SAS® IT Resource Management 2.7: Macro Reference

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS® IT Resource Management 2.7: Macro Reference*. Cary, NC: SAS Institute Inc.

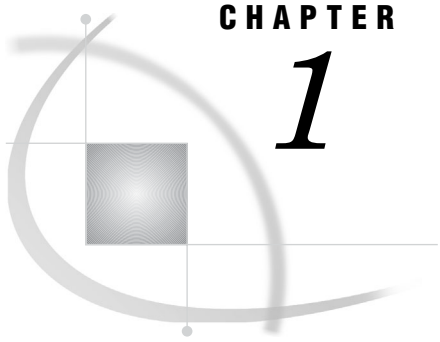
SAS® IT Resource Management 2.7: Macro Reference

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, July 2004

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Contents



CHAPTER

1

Introduction

Overview of SAS IT Resource Management Macros

SAS IT Resource Management provides interactive and batch facilities with which you can access, manage, analyze, and present your performance data. The SAS IT Resource Management batch macros provide a batch method of performing many of the same functions that you can perform within the SAS IT Resource Management GUI. For example, there are macros for

- starting SAS IT Resource Management
- processing data
- reducing data
- managing your data dictionary
- creating reports.

You can write a batch script or program that contains calls to SAS IT Resource Management macros and then submit the script as a job to be processed through the batch processing facility on your system. You can also submit and run the macros through the SAS Program Editor window. (For more information about submitting batch jobs, see the topic "Work with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide* and the examples in this chapter.) The macros run outside the SAS IT Resource Management GUI.

Within the SAS IT Resource Management GUI, you can also perform tasks and save the macro source code for those tasks. You can then run the source code in batch mode.

The term *batch* commonly refers to any noninteractive job that is scheduled to run at a later time. That time could be a few minutes later, and therefore the job could run in

parallel with your current interactive tasks. The job could also be scheduled to run overnight.

You can create a batch script by saving to a file the macro source for the commands that you select through the user interface. You can also write the batch script from scratch by using the SAS IT Resource Management macros. Also, if you are running SAS IT Resource Management server software on z/OS, then you can schedule a file that you made by using the SAS IT Resource Management z/OS Batch Utility (see "Example: The SAS IT Resource Management z/OS Batch Scheduler" on page 19).

Executing Macros in Batch Mode

You can submit and run SAS IT Resource Management macros in batch mode in UNIX, z/OS, and Windows environments. The term *batch* commonly refers to jobs that are run in the background, but for some environments, such as UNIX, it can also refer to a collection of jobs that are submitted to run together. In this document, batch refers to jobs that run in the background.

To run macros in batch mode, you need to create a batch file that contains calls to one or more macros. You can then submit the file and the job will be processed in batch mode on your system. You can develop and code the batch file independent of the SAS IT Resource Management graphical user interface, or you can create the macro code by selecting tasks through the software interface and by saving the underlying macros (SOURCE) in a file. If you create the file by saving options that are selected through the SAS IT Resource Management GUI, then you might need to edit the batch file to include additional commands specific to your job, such as SAS, SAS IT Resource Management, or local system statements and options. If the batch job is running on z/OS, then you should also include the appropriate JCL statements. You can then submit the batch job for background processing.

For a sample batch file or job for your operating environment, see the following examples:

- "Example: Creating a Batch File for UNIX" on page 17
- "Example: Creating a Batch File for Windows" on page 16
- "Example: Creating a Batch Job for z/OS" on page 11
- "Example: The SAS IT Resource Management z/OS Batch Scheduler" on page 19

If you want to run these batch programs on a regular basis, then you can schedule jobs to execute automatically by using operating environment utilities, such as **`cron`** on UNIX, the AT command on Windows, or a batch scheduler on z/OS.

Refer to the SAS Companion for your operating environment in order to find more information about scheduling and running batch jobs and specifying SAS system options.

Note: If you create your own SAS macros or macro variables, then avoid names that start with the letter pairs CM, CP, CV, CS, CV, or CW, unless they are specifically shown in the documentation. SAS IT Resource Management macro names and macro variable names start with these same letter pairs, and unpredictable results might occur if you use these "reserved" macro names for your own macros and macro variables. △

For more information on using macros, see the following topics:

- For a complete list of SAS IT Resource Management macros, refer to "SAS IT Resource Management Macros Grouped by Task" on page 22.
- For information on what to do when batch jobs fail to execute, see "Troubleshooting Batch Jobs That Fail" on page 655. Look for information on the specific macro that you are using.

- For information on macro document conventions, see “Macro and Syntax Document Conventions” on page 3.

Macro and Syntax Document Conventions

Each macro description provides the macro name, the macro overview, the syntax (how to call the macro), information about the parameters, and any notes that relate to using the macro. When the macro name begins with the letters CP, the macro is usually available for UNIX, Windows, and z/OS operating environments. The letters CS at the front of the macro name indicate that the macro applies only to the UNIX operating environment; the letters CW indicate Windows environments; and the letters CM indicate the z/OS operating environment.

In a few instances, the macro name begins with the letters CP, but the macro is not currently available in all three operating environments. In these cases, the macro description displays the platform names in order to indicate the operating environment in which the macro is available.

When a macro is available in two or more operating environments, some of the parameters might not be available on the different hosts, might function differently on each host, or might have host-specific values. In these situations, the host-specific differences are identified by host-specific icons and are described within the parameter definition.

The “Syntax” section of each macro description lists the parameters that you can use with the macro. Macros can use the following types of parameters:

positional parameters
: are identified by their position in the macro invocation. Positional parameters are not specified using a keyword; the macro understands the purpose of the parameter because the parameter occurs in a specific position in the calling sequence. In the macro syntax, positional parameters are displayed in italics. All positional parameters precede keyword parameters and should be specified in the order in which they appear in the macro syntax.

  The following example macro has three positional parameters and two keyword parameters. It will be used throughout this topic:

```
%macro example(pos1, pos2, pos3, keyword1=, keyword2= );
    %put pos1=&pos1;
    %put pos1=&pos1;
    %put pos1=&pos1;
    %put keyword1=&keyword1;
    %put keyword2=%keyword2;
    %mend;
```

  If you want to specify any positional parameter, then you must use a comma as a placeholder for all preceding unspecified positional parameters, as shown in the following example:

```
%example(,,value_for_pos3);
```

  The next example is incorrect. If you fail to use commas as placeholders for positional parameters 1 and 2, then the value that is specified as positional parameter will be assigned to positional parameter one.

```
%example(value_for_pos3);
```

  You must specify a comma after the last positional parameter if it is followed by keyword parameters. However, you do not need to

specify multiple commas as placeholders for any nonspecified positional parameters that fall between the last specified positional parameter and the first keyword parameter.

In the following example, the last two commas are not required, but including them will not result in errors:

```
%example(value_for_pos1,,);
```

Likewise, in the following example, the last two commas are not required, but including them will not result in errors:

```
%example(value_for_pos1,,,keyword1=value1);
```

In the following example, commas are not required for positional parameters 2 and 3, and therefore commas are not used to take the place of those positional parameters:

```
%example(value_for_pos1,keyword1=value1);
```

However, if you do not specify any positional parameters before you specify your first keyword parameter, then you MUST NOT specify a comma. This case is shown in the following example:

```
%example(keyword1=value1);
```

The following example shows a case where you do not specify any positional parameters and where you specify the optional keyword parameters in random order:

```
%example(keyword4=value4, keyword2=value2, keyword5=value5);
```

If you do not specify any positional parameters, then no comma is required at all:

```
%example();
```

keyword parameters

begin with a specific keyword, followed by an equal sign and then a value. You can specify these parameters in any order that you want, but they must be specified after any positional parameters. If a parameter has a default value, then this value is identified in the description of the parameter. *A comma is required to separate more than one keyword parameter, or to separate keyword parameter(s) from preceding positional parameters. However, if the macro does not have positional parameters or if you do not specify any positional parameters before your first keyword parameter, then you MUST NOT specify a comma before the first keyword parameter or the macro will fail.* If you use both positional and keyword parameters, then place a comma after the last positional parameter and between the keyword parameters:

```
%example(,value_for_pos2, keyword1=value1, keyword2=value2);
```

If you do not use positional parameters, then you MUST NOT use a preceding comma before the first keyword parameter:

```
%example(keyword1=value1, keyword2=value2);
```

Omitted keyword parameters do not need placeholder commas:

```
%example(keyword2=value2);
```

Some keyword parameters are required. Required keyword parameters are listed in alphabetical order immediately following the positional parameters.

Keyword parameters that are not required are listed in alphabetical order after the required keyword parameters in the "Syntax" section of each macro. Optional keyword parameters are enclosed in angle brackets, as shown in the syntax example below.

Suggested Typography

Angle brackets are used to indicate optional parameters, as in <optional parameter>. Commas are displayed within the angle brackets to indicate that the comma is required only if the next parameter is specified.

For positional parameters, a comma might be required depending on whether additional positional parameters are specified following the initial positional parameter.

For keyword parameters, a comma is required preceding each keyword parameter.

In this example, assuming all parameters are optional, the syntax is documented as follows:

```
%example(  <pos1>
           ,<pos2>
           ,<pos3>
           <,optional-keyword1=value1>
           <,optional-keyword2=value2> );
```

A footnote can be included in a parameter description in order to indicate when you can omit a trailing comma after the last specified positional parameter. However, you can specify

```
%example(value_for_pos1,,,keyword1=value1);
```

instead of the simpler

```
%example(value_for_pos1,keyword1=value1);
```

and either way is correct.

Type styles and some symbols have special meanings when they are used within the macro syntax. The following list explains the style conventions that are used for syntax within this document:

UPPERCASE ROMAN	identifies macro names, keyword parameters, and values that are literals.
italics	identifies positional parameters or values that you supply.
| (vertical bar)	indicates that you can choose one value from a group. Parameters or values that are separated by bars are mutually exclusive.
... (ellipsis)	indicates that the arguments or group of arguments that follow the ellipsis can be repeated any number of times.
< > (angle brackets)	identifies optional parameters for the macro. Any parameter that is not enclosed in angle brackets is required.

Terminology

The following terms are used in macro descriptions to indicate parameter values that you supply:

physical-filename — the physical location and name of a file or library. This includes the complete directory path or high-level qualifiers and the name of the file or member. For example, on UNIX you would specify the *physical-filename* value in a format such as /sas/cpe/filename; in Windows environments you would specify it in a format such as c:\sas\cpe\filename; and on z/OS you would specify it in a format such as SAS.CPE(MEMBER).

pdb-name — the complete physical location of the partitioned database (PDB). This includes the complete directory path or high-level qualifiers and the PDB name. It does not include the specific library name, such as DETAIL, DAY, or WEEK. For example, the PDB= parameter in the %CPSTART macro identifies the active PDB. On UNIX you would specify the *pdb-name* value in a format such as /sas/cpe/pdbname; in Windows environments you would specify it in a format such as c:\sas\cpe\pdbname; and on z/OS you would specify it in a format such as SAS.CPE.PDBNAME.

If the PDB name is omitted, then SAS IT Resource Management uses the PDB that was most recently accessed.

root-location — the physical location where the PGMLIB library is installed in your operating environment. This includes the directory path or high-level qualifiers where PGMLIB is installed, but it does not include the library name. For example, in the %CPSTART macro, you might specify the *root-location* value for the ROOT= parameter as /sas/cpe on UNIX, as c:\sas\cpe in Windows environments, and as SAS.CPE on z/OS.

site-library — the physical location of the SAS IT Resource Management site library in your operating environment. This includes the directory path or high-level qualifiers and the name of the site library. The site library contains system options and global information for your site. For example, in the %CPSTART macro, you might specify the *site-library* value for the SITELIB= parameter as /sas/cpe/sitelib on UNIX, as c:\sas\cpe\sitelib in Windows environments, and as SAS.CPE.SITELIB on z/OS.

Global and Local Macro Variables

SAS IT Resource Management provides global and local macro variables. Macro variables have built-in defaults, but you can assign a different value to a variable in order to globally control processing and simplify tasks.

Global Macro Variables

There are two types of global macro variables: non-persistent and persistent. The following global macro variables are non-persistent:

- CPAUTOFT
- CPFUTURE
- CPLRMAX
- CPOPSYS
- CPWSTYLE
- CPXPDBNM
- CPYVAL.

If you assign a new value to one of these non-persistent global macro variables during an interactive or batch session, then the new value applies only during that SAS session. The new value is not saved.

The following global macro variables are persistent:

- CPBEGIN
- CPEND
- CPREDLVL
- CPWHERE.

During an interactive session, if you assign a new value to one of these persistent global macro variables, then the new value applies during that SAS session, the value is saved (in your SASUSER library), and the value becomes the new default value for interactive and batch sessions.

During a batch session, if you assign a new value to one of these persistent global macro variables, then the new value applies only during that SAS session. The value is not saved.

You can change the value of a global macro variable at any time. You can also set the value to missing, as in:

```
%let CPBEGIN=;
```

To set global macro variables in a batch job, use the following format:

```
%let macro-variable = value;
```

For example:

```
%let cpwhere = 'host'
;
```

CPBEGIN sets the initial date and time so that you can subset data by a datetime range. Specify this value by using one of the following formats: *ddmmmyy:hh:mm* or *ddmmmyyyy:hh:mm*. For example, you would specify 1:30 p.m. on September 1, 2003, as 01SEP03:13:30. You can also use the TODAY*[-nnnn unit]*@*hh:mm]* format, which uses the current date value, minus *nnnn* units, at the specified hour. (Units can be day(s), week(s), month(s), or year(s). The default unit is days.) You can use this macro variable to set a begin time prior to the current date. You can also use this variable with the CPEND variable to set a datetime range.

If you do not use this macro variable, then the default value is the date and time of the earliest observation in the specified level of the specified table, as stored in the PDB's data dictionary or as obtained by scanning the data.

If you set begin and end dates in your report definition, then those dates will override any global dates. If you do not set begin and end dates in your report definition and you have global begin and end dates, then the global dates are used.

CPEND sets the final date and time so that you can subset data by a datetime range. Use the same datetime format as used for CPBEGIN.

If you do not specify this macro variable, then the default value is the date and time of the latest observation in the specified level of the specified table, as stored in the PDB's data dictionary or as obtained by scanning the data.

If you set begin and end dates in a report definition, then those values will override any global dates. If you do not set begin and end dates in your report definition, then the global dates are used.

CPFUTURE controls the processing of incoming data whose DATETIME variable contains a value that is more than 48 hours in the future from the current system time. (For details that are related to your specific process macro and data source, see the %CxPROCES details that follow.) The current system time is the current time on the system where data is being staged for processing into the PDB. The 48-hour buffer allows for different time zones, daylight saving time, Greenwich Mean Time, and so on.

Note: %CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES. △

The %CxPROCES macros use the CPFUTURE macro variable as follows:

%CMPROCES and %CPPROCES macros
use the CPFUTURE macro variable when you process data. The default value of the CPFUTURE macro variable is used (CPFUTURE=DISCARD) unless you specify another value for CPFUTURE.

%CSPROCES and %CWPROCES macros
use the CPFUTURE macro variable if, and only if, the COLLECTR= parameter is set to GENERIC and the TOOLNM= parameter is set to SASDS. In this case, the MINDATE= and MAXDATE= parameters are ignored.

%CSPROCES and %CWPROCES macros
ignore the CPFUTURE macro variable if the value for COLLECTR= is not GENERIC or if the value of TOOLNM= is not SASDS. In this case, the MINDATE= and MAXDATE= parameters on %CSPROCES or %CWPROCES are used.

Possible values for the CPFUTURE macro variable are as follows:

DISCARD Any data with a DATETIME value 48 or more hours in the future is not staged for processing and is not processed into the PDB. A warning is written to the SAS log, notifying you that future data was encountered. All other data in the raw data file is staged for processing and is processed into the PDB. This value of CPFUTURE is used to prevent future data from being processed into the PDB. The future data might cause existing

data to be aged out (the existing data would appear to be older than it is, in comparison with the future data). *This is the default.*

ACCEPT — All incoming data is staged for processing and processed into the PDB. If any of the data has a DATETIME value 48 or more hours in the future, then a note that future data was encountered is written to the SAS log. This value of CPFUTURE is used to enable a PDB to accept future data. For example, you might want to use this setting to perform end-of-year testing with a test PDB.

Note that age limits take effect from the most recent data, so dates in the future might cause at least some of the existing data to be aged out of the PDB.

TERMINATE — If any incoming data has a DATETIME value 48 or more hours in the future, then staging of the data stops, an error message is written to the SAS log, and the macro terminates. Neither the future data nor any other raw data that is in front of it or behind it in the raw data file is processed into the PDB. This value of CPFUTURE is used to prevent future data from being processed into the PDB, which might cause existing data to be aged out (the existing data would appear to be older than it is, in comparison with the future data). This value stops processing and thus calls more attention to the future data than to the DISCARD value.

If you want to use a value for CPFUTURE other than the default, then you can specify the new value with a %LET statement:

```
%let CPFUTURE=new_value;
```

In batch mode, the %LET statement must occur before the %CxPROCES macro. If you are using the SAS IT Resource Management GUI, then type the %LET statement in the SAS Program Editor window and submit the contents of that window for processing.

To allow future data, you can set CPFUTURE=ACCEPT. This enables you to process data with "future" datetimes. Within the SAS IT Resource Management GUI, you can submit the %LET statement from the command line or the SAS Program Editor window. In batch, you would submit this statement before you submit the %CxPROCES macro.

CPLRMAX — keeps track of the highest value of the return code from %CPLOGRC during the current SAS session. The value of CPLRMAX is not reset by the START code in the call to %CPLOGRC. Thus, at the end of the SAS session, you can use the value of CPLRMAX as one measure of success over all the %CPLOGRC sequences in the SAS session. For more information, see the macro "%CPLOGRC" on page 128.

CPOPSYS — indicates the operating environment in which you are running SAS IT Resource Management. This macro variable is initially set by the

%CPSTART macro. The default is specific to the operating environment. The one case in which you might want to change the value of this macro variable occurs when you are running on z/OS and you want to send Web-enabled output to z/OS directories. For more information about z/OS directories, which are provided in z/OS by the z/OS UNIX File System, see the SAS Companion for z/OS.

To set this variable, specify the following:

```
%let cpopsys=OSYS;
```

The next call to the %CPSTART macro restores the original value of CPOPSYS. To reset the value to z/OS without using %CPSTART, specify the following:

```
%LET cpopsys=MVS;
```

Specifying CPOPSYS=OSYS enables you to use directory names on z/OS. For example, you can specify a UNIX directory name in the HTMLDIR= and IMAGEDIR= parameters on many of the report macros when you are creating reports to display on the Web, or in the DIR= parameter on the %CMFTPSND macro.

In batch mode, the %LET statement must occur after the %CPSTART macro and before calls to any macros that generate Web-related output to z/OS directories. In the SAS IT Resource Management GUI (after you invoke SAS IT Resource Management and before you generate Web-related output to z/OS directories), type the %LET statement in the SAS Program Editor window and then submit the contents of that window for processing.

CPWHERE when reporting on data, puts data into subsets with a logical WHERE statement. Use the standard format for SAS WHERE expressions. For more information, see WHERE expressions in the *SAS Language Reference* documentation for your current release of SAS.

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. △

The following examples illustrate valid WHERE expressions:

```
%let cpwhere=machine='host1';
%let cpwhere=machine in ('host1','host2','host3');
%let cpwhere=machine is not missing;
%let cpwhere=machine is like 'host%';
%let cpwhere=machine contains 'host';
%let cpwhere=errors > 200;
%let cpwhere=datetime < '01jan2003:00:00'dt;
```

If you do not specify this macro variable, then a global WHERE statement is not used.

CPWSTYLE replaces the default value for the WEBSTYLE= parameter. The valid values are GALLERY2 and DYNAMIC.

The specified value of CPWSTYLE takes precedence over the default value for the WEBSTYLE= parameter. Thus, if the WEBSTYLE= parameter *is not* specified in a call to a macro that can have the WEBSTYLE= parameter (%CPCCHRT, %CPCHART, %CPG3D, %CPHTREE, %CPMANRPT, %CPPLOT1, %CPPLOT2,

%CPPRINT, %CPRUNRPT, %CPSPEC, %CPSRCRPT, %CPTABRPT, %CPWEBINI, and %CPXHTML), then the style is specified by CPWSTYLE. For example, if you add the code

```
%let CPWSTYLE=DYNAMIC;
```

to a program before the program encounters a call to %CPRUNRPT that does not have a WEBSTYLE= parameter, then the call operates as if it has WEBSTYLE= set to DYNAMIC.

Note: The global macro variable CPWSTYLE has *no* effect on a macro call that specifies the WEBSTYLE= parameter. △

CPXPDBNM (z/OS only) indicates whether additional views are to be created so that you can run old reports that were originally designed to run against the MXG PDB. For more information on setting the value of CPXPDBNM, see "%CPPDBOPT" on page 135. Also see the section "Viewing/Editing Your Active PDB's MXG Views Property" in the chapter "Administration: Working with Whole PDBs" in the *SAS IT Resource Management User's Guide* and the section "Viewing/Editing a Site Library's MXG Views Property" in the chapter "Administration: Working with Site Libraries" in the *SAS IT Resource Management User's Guide*.

CPYVAL sets the default value for the YVAL parameter if none is specified by the user. An example of its use follows: if you want to generate reports with 24 bars, corresponding to each hour of the day, then set the following after CPSTART but before any report macros

```
%let CPYVAL=24;
```

Local Macro Variables

Local macro variables are set individually for each macro when you invoke the macro. These variables are described within each macro description and syntax. For more information on a specific macro variable, refer to the appropriate macro.

You set the variables in batch mode by specifying the parameters on the macro, such as BEGIN= and END=. These variables temporarily override the values of the global macro variables, except when you specify a WHERE expression that begins with the key phrase SAME AND. In this case, the local WHERE expression is temporarily appended to the global WHERE expression. (For more information on WHERE expressions, see WHERE expressions in the *SAS Language Reference* documentation for your current release of SAS.)

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. △

Example: Creating a Batch Job for z/OS

The program in the following example performs daily processing and reduction on a PDB. At any time, you can modify a batch job by adding, deleting, or changing macros so that the program performs the tasks that you want. Many of the macro descriptions also contain examples that are specific to that macro.

Note: For similar job that is machine-readable, see the CMJCLLD member in the CPMISC PDS that was set up during installation at your site. For the full name of this

PDS at your site, see your site administrator or your SAS Installation Representative. The root location refers to the high-level qualifiers where SAS IT Resource Management is installed at your site. △

```
//DAILY   JOB (accounting
info),'Daily job',
//       MSGLEVEL=(1,1),TIME=20,REGION=32M,NOTIFY=
//*****************************************************************
//*                                                               *
//* This SAS IT Resource Management for z/OS                      *
//* example runs a daily job that                                 *
//* processes new data from SMF log file(s) and adds it to the    *
//* DETAIL level of the PDB.  It reduces data from detail level   *
//* into the DAY, WEEK, MONTH, and YEAR reduction levels.         *
//*                                                               *
//* This job now also includes a step to back up your PDB by using *
//* IBM's DFDSS program. You can use the DFDSS DUMP command to back *
//* up volumes and data sets, and you can then use the RESTORE    *
//* command to recover them. You can also make incremental backups *
//* of your data sets by doing a data set DUMP with RESET specified *
//* and filtering on the data-set-changed flag.                   *
//* For more information on this facility reference the IBM manual: *
//* Data Facility Data Set Services: Reference V2R5 (SC26-4389-03). *
//* You may want to update that step with the backup procedure    *
//* commonly used at your site instead.                           *
//*                                                               *
//*===============================================================*
//*                                                               *
//* NOTE:  This job requires that the PDB already exist.          *
//*        If you need to allocate a PDB first,                   *
//*        see the member CMPDBALC in this CPMISC PDS.            *
//*                                                               *
//* This job requires the following customization:                *
//*                                                               *
//*  1) Change the JOB card to a valid job card for your system.  *
//*                                                               *
//*  2) Do a global change of YOUR.SMF.DATA to the data set       *
//*     name of your SMF data sets.                               *
//*                                                               *
//*  3) Do a global change of MXG.USERID.SOURCLIB to the name     *
//*     of your customized MXG source library.                    *
//*                                                               *
//*  4) Do a global change of MXG.MXG.SOURCLIB to the name of the *
//*     PDS containing the original MXG source library.           *
//*                                                               *
//*  5) Do a global change of MXG.MXG.FORMATS to the name         *
//*     of your MXG format library.                               *
//*                                                               *
//*  6) Do a global change of YOUR.ITRM to the high-level         *
//*     qualifiers of this application on your z/OS system.       *
//*     Make sure this qualifier has been set correctly for       *
//*     CONFIG= on the LOAD job step.                             *
//*                                                               *
//*  7) Do a global change of YOUR.ITRM.PDB to the high-level     *
//*     qualifier for your PDB. This PDB should have one or more  *
```

```
//*     tables already in it.                                    *
//*                                                              *
//* Plus, in the BACKUP step using IBM's ADRDSSU program:        *
//*                                                              *
//*  8) Modify characteristics on the DD named TAPE allocation   *
//*     and/or on the DUMP command according to your needs or    *
//*     site-specific rules. For instance, you may want to create*
//*     and use GDGs for the backup tape data sets.              *
//*                                                              *
//*  9) ADRDSSU may not work when any of the PDB's SAS data      *
//*     libraries span multiple volumes.  For more information,  *
//*     check with SAS Technical Support.                        *
//*                                                              *
//****************************************************************
//BACKUP  EXEC PGM=ADRDSSU
//TAPE     DD DSN=your.itrm.pdb,DISP=(OLD,KEEP),
//            UNIT=cart,RETPD=30
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *
  DUMP DATASET(INCLUDE(your.itrm.pdb.*)) OUTDD(TAPE) -
       SHARE COMPRESS
//*
//LOAD    EXEC SAS,WORK='10500,2000',
//        CONFIG='your.itrm.cpmisc(CMCONFIG)'
//SMF      DD DSN=your.smf.data,DISP=SHR
//SYSIN    DD DATA,DLM='$$'

*-------------------------------------------------------------------*
  * This application will not run unless %CPSTART has been executed *
  * %CPSTART allocates this application and the PDB.                *
  * The MXGLIB= and MXGSRC= parameters are required for the         *
  * MXG software tool used  within %CMPROCES below.                 *
  * The ROOTSERV= and SHARE= parameters should only be used if      *
  * SAS/SHARE  is being used with this application.                 *
  * If the _RC= parameter is nonzero from %CPSTART,                 *
  * check the explanatory message in the SAS log.                   *
  *                                                                 *
  * The input stream should begin with a call to the %CPSTART       *
  * macro, with MODE=BATCH specified. Then, the input stream can    *
  * have SAS statements, other calls to SAS IT Resource Management  *
  * macros, etc.                                                    *
  *-------------------------------------------------------------------* ;


  /* verify root= pdb= mxglib= mxgsrc= in the following macro */
  %CPSTART(MODE=BATCH,
           ROOT='your.itrm.',
           ROOTSERV=,
           PDB='your.itrm.pdb.',
           DISP=OLD,
           SHARE=N/A ,
           MXGLIB=mxg.mxg.formats,
           MXGSRC=('mxg.userid.sourclib' 'mxg.mxg.sourclib'),
           _RC=cpstrc
```

```
        );

%PUT CPSTART return code is &cpstrc;


*------------------------------------------------------------*
* The %CMPROCES macro processes data into the PDB that was   *
* established by the %CPSTART macro above.                   *
* As no table list is specified as the second parameter all  *
* the tables in the PDB for this collector with a table kept *
* indicator of YES will be processed.                        *
* TOOLNM=MXG and COLLECTR=SMF parameters must be specified.  *
* If the _RC= parameter is nonzero from %CMPROCES,           *
* check the explanatory message in the SAS log.              *
*------------------------------------------------------------* ;


%CMPROCES(,
          COLLECTR=SMF,
          TOOLNM=MXG,
          _RC=cmprrc
         );

%PUT CMPROCES return code is &cmprrc;



*------------------------------------------------------------*
* The %CPREDUCE macro reduces data in the active PDB detail  *
* level into the day, week, month and year reduction levels. *
* The variables and statistics kept are defined in the data  *
* dictionary.                                                *
* As no table list is specified as the first parameter,      *
* reduction will be performed on all the tables in the PDB.  *
* If the _RC= parameter is nonzero from %CPREDUCE            *
* check the explanatory message in the SAS log.              *
*------------------------------------------------------------* ;


%CPREDUCE(,_RC=cpredrc);

%PUT CPREDUCE return code is &cpredrc;

$$
//
```

Processing Notes

1. Before you submit the job for the first time, you must perform the following steps, typically from the graphical user interface within the application:
   a. Estimate the space requirements for your PDB and allocate space accordingly.
   b. Add table definitions to the PDB, as required. If the table definitions that are explicitly specified by a table list in the %CMPROCES macro are not in the

PDB, then the %CMPROCES macro automatically adds those table definitions to the PDB by copying them from the master data dictionary.

c Edit the table definitions to adjust durations for retention of DETAIL, DAY, WEEK, MONTH, and YEAR data, as required.

d Add, update, or delete variables in the tables, as required.

e Edit the variable statistics to adjust which statistics are requested for DAY, WEEK, MONTH, and YEAR levels, as required.

f Verify that all tables and variables that you want to use are marked KEPT=YES.

2 The %CPSTART macro invokes SAS IT Resource Management and allocates the PDB.

The MODE= parameter must be set to *batch*.

The ROOT= parameter specifies the high-level qualifiers in the data set names for this application. For example, if SAS IT Resource Management is located in a SAS library that is installed at your site with the name *your.itrm.pgmlib*, then use *your.itrm* as the value for the ROOT= parameter. Similarly, the PDB= parameter specifies the high-level qualifiers in the z/OS data set names for the SAS data libraries in the PDB. For example, if the PDB is in libraries that are all named *your.itrm.pdbname.library*, where *library* varies depending on the library name (such as DETAIL, DAY, WEEK, MONTH, and YEAR), then use *your.itrm.pdbname* as the value for the PDB= parameter.

The MXGLIB= parameter specifies the fully qualified name of the SAS library that contains the SAS formats that are provided by MXG. The MXGSRC= parameter specifies fully qualified names of one or more PDSs that contain MXG SOURCLIB source entries and user overrides to these source entries. List the PDSs in search order. If the same member name exists in more than one of the PDSs, then the member that is used is the first one that is encountered. Thus, typically, list the PDS that has your source entries in front of the PDS with the MXG source entries.

3 If the batch job contains the %CMPROCES macro, then you must specify the name of the z/OS data set that contains the logged SMF-style data in one of the following ways:

- with a DD statement
- with a FILENAME statement, such as

```
filename smf SMF_style_log_name;
```

  If you use the FILENAME statement, then place it before the %CMPROCES macro invocation.

- as the first positional parameter in the %CMPROCES macro.

4 The %CMPROCES macro reads logged SMF data from the log file and processes it into the detail level of the PDB for the specified tables. Specifying a list of table names is optional. If no names are specified, then the macro processes all tables in your PDB with KEPT=YES.

The PDB into which the data is processed is the PDB that is specified in the %CPSTART macro by the PDB= parameter. This is referred to as the active PDB.

The TOOLNM= parameter in the %CMPROCES macro can be set to MXG or SASDS. This application uses the SAS library and the PDSs that are specified in the %CPSTART macro by the MXGLIB= and MXGSRC= parameters.

5 The %CPREDUCE macro reduces the data in the DETAIL level of the active PDB into the DAY, WEEK, MONTH, and YEAR reduction levels. Specifying a table name is optional. If no table name is specified, then the macro reduces all tables in your PDB that have KEPT=YES and the AGELIMIT value > 0 in at least one of

the DAY, WEEK, MONTH, or YEAR levels. The *level* must also have at least one variable with a selected statistic and KEPT=YES.

Example: Creating a Batch File for Windows

The .bat file in the following example starts SAS in batch mode from the DOS command line. The comments within the code describe how to use the script. Many of the macro descriptions also contain examples that are specific to that macro.

For more information about setting up your daily process and reduction job(s) or your data collector, see the sections "General-Purpose Server Setup Documentation" and "Collector-Specific Setup Documentation" in the section "Locating Help" in "Chapter 1: Introduction to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

```
@echo off
Rem - -- - - - - - - - - - - - - - - - - - - - - - - - - - -- -- -- --
Rem Copyright (c) 2004 by SAS Institute Inc., Cary, NC 27513
Rem Name:  sasbat.bat
Rem Doc: Run SAS on Windows with proper options for SAS IT
Rem      Resource Management batch mode
Rem
Rem Usage: sasbat.bat infile
Rem    where "infile" is the simple (path is not required
Rem    if in the current working directory)
Rem    first name of the files read and written by SAS:
Rem       .sas is appended to "infile" for the input program
Rem       .log is appended to "infile" for the SAS log
Rem Example: cd \mypgms
Rem          sasbat.bat mysaspgm
Rem - -- - - - - - - - - - - - - - - - - - - - - - - - - - -- -- -- --
set INFILE=%1%

Rem The following line is broken into three lines so that you can
Rem  see the right end of the command. Make these three lines into
Rem  a single line before you run.
"C:\program files\sas\sas 9.1\sas.exe"
    -sysin %INFILE% -awstitle "SAS ITRM Batch" -icon
    -nosplash -noxwait -noterminal


Rem Note common exit codes
Rem The following line is broken into two lines so that you can
Rem  see the right end of the command. Make these two lines into
Rem  a single line before you run.
for %%c in (0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16)
    do if errorlevel %%c set EXITCODE=%%c

Rem Did SAS terminate gracefully?
Rem The following line is broken into two lines so that you can
Rem  see the right end of the command. Make these two lines into
Rem  a single line before you run.
find "NOTE: SAS Institute Inc., SAS Campus Drive"
      %INFILE%.log > nul
```

```
if not errorlevel 1 goto NORMTERM
echo SAS was interrupted before completing termination processing.
goto EXIT

:NORMTERM

Rem Go scan log when exitcode is 0 or 1.
if %EXITCODE%==0 goto SCANLOG
if %EXITCODE%==1 goto SCANLOG

:ERRORS
echo SAS terminated with errors (EXITCODE ge %EXITCODE%).
goto EXIT

:SCANLOG
Rem Scan the SAS log since we might still have errors with
Rem  zero exitcode.
find "Errors printed on page" %INFILE%.log > nul
if not errorlevel 1 goto ERRORS
echo SAS terminated without errors (EXITCODE eq %EXITCODE%).
goto EXIT

:EXIT
%echo on
```

Example: Creating a Batch File for UNIX

The UNIX shell script in the following example starts SAS in batch mode from the UNIX command line and displays a message that is based on the return code from the SAS session. The comments within the code describe how to use the script. Many of the macro descriptions also contain examples that are specific to that macro.

For more information on setting up your daily process and reduction jobs or your data collector, see "Part 2: Setup" in the *SAS IT Resource Management User's Guide* and also follow any references there to the *SAS IT Resource Management Server Setup Guide*.

```
#!/bin/ksh
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+
# Copyright (c) 2004 by SAS Institute Inc., Cary, NC 27513
# Name:    sasscript.sh
# Doc:     Runs SAS with proper options for SAS IT Resource
#          Management batch mode
#
# The input file contains the job code.
# - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -+

PROGNAME='sas '

# - - - - - - - - - - - - - - - - - - - +
# First, be helpful
# - - - - - - - - - - - - - - - - - - - +
if test "$#" = "0"
   then echo " "
        echo "Usage: $PROGNAME (sasfile)"
```

```
        echo " "
        echo "where 'sasfile' is the simple (no dots, no dirname)' "
        echo "first name of the files read and written by SAS:"
        echo " "
        echo "  .sas is appended to 'sasfile' for the input program"
        echo "  .log is appended to 'sasfile' for the output SAS log"
        echo " "
        echo "Example: cd/u/myid/mypgms"
        echo "         sasscript.sh mysasprog"
        echo " "
        echo "This reads the SAS program in /u/myid/mypgms/mysasprog.sas,"
        echo "and creates a SAS log in /u/myid/mypgms/mysasprog.log."
        exit 0
        fi

# - - - - - - - - - - - - - - - - - - - - +
# Begin
# - - - - - - - - - - - - - - - - - - - - +
# In your script, confirm that both of the +
# single quotes around date go from upper  +
# left to lower right, so that             +
# date will be evaluated.                  +
echo $PROGNAME: Starting at 'date'

 INFILE=$1

rm $INFILE.log 2> /dev/null

# - - - - - - - - - - - - - - - - - - - - +
# Start SAS
               sas -noterminal $INFILE.sas

# - - - - - - - - - - - - - - - - - - - - +
# Advertise exit code
# - - - - - - - - - - - - - - - - - - - - +
EXITCODE=$?

# Did SAS terminate gracefully?
grep "NOTE: SAS Institute Inc., SAS Campus Drive" $INFILE.log > /dev/null
if test $? -ne 0
   then echo SAS was interrupted before completing termination processing.
fi

# Complain appropriately based on the exit code.
if test $EXITCODE -lt 2
   then # We can still have errors even with exit code of 0 or 1
        grep "Errors printed on page" $INFILE.log > /dev/null
        if test $? -eq 0
           then echo "SAS terminated with errors."
           else echo "SAS terminated without errors."
        fi
   else echo "SAS terminated with errors: EXITCODE eq $EXITCODE"
fi
```

```
# - - - - - - - - - - - - - - - - - - - +
# That is all
# - - - - - - - - - - - - - - - - - - - +
# In your script, confirm that both of the +
# single quotes around date go from upper  +
# left to lower right, so that             +
# date will be evaluated.                  +
echo $PROGNAME: Ending at 'date'

exit $EXITCODE
```

Example: The SAS IT Resource Management z/OS Batch Scheduler

If you use SAS IT Resource Management with z/OS, then you can use the Batch Utility, which is available from the server interface on z/OS, in order to save and schedule jobs to be processed. You can then submit the job to run in batch mode by creating a file that calls the %CPBATCH macro. For example, if you run the job on a Tuesday, then the Batch Utility runs all the tasks with a schedule of Daily, all the tasks with a schedule of Weekday, and all the tasks with a schedule of Tuesday. The order in which it runs the tasks is determined by the priorities that are set in the Batch Utility. For more information about the z/OS batch scheduling subsystem, refer to the Batch Help Index that is available from the server interface on z/OS.

The following job submits the tasks that were set up by using the Batch Utility. The comments within the program describe areas of the code that you should change in order to customize the code.

Note: For a similar job that is machine-readable, see the CMJCLSCD member in the CPMISC PDS that was set up during installation at your site. For the full name of this PDS at your site, see your site administrator or your SAS Installation Representative. The root location refers to the high-level qualifiers where SAS IT Resource Management is installed at your site. △

```
//SCHEDULE JOB (accounting
info),'schedule job',
//       MSGLEVEL=(1,1),TIME=20,REGION=32M,NOTIFY=
//*****************************************************************
//*                                                               *
//* This SAS IT Resource Management for z/OS                      *
//* example runs a daily job.                                     *
//* It runs the list of tasks that have previously been           *
//* set up by your site administrator.                            *
//* To set up the schedule, use the Batch Utility                 *
//* within the SAS IT Resource Management application.            *
//* Schedule the job to run after the nightly SMF dump.           *
//*                                                               *
//*===============================================================*
//*                                                               *
//* NOTE:  This job requires that the PDB already exist.          *
//*        If you need to allocate a PDB first,                   *
//*        see the member CMPDBALC in the CPMISC pds.             *
//*                                                               *
//* This job requires the following customization:                *
//*                                                               *
//*  1) Change the JOB card to a valid job card for your system.  *
```

```
//*                                                                   *
//*  2) Do a global change of YOUR.SMF.DATA to the data set           *
//*     name of your SMF data sets.                                   *
//*                                                                   *
//*  3) Do a global change of MXG.USERID.SOURCLIB to the name         *
//*     of your customized MXG source library.                        *
//*                                                                   *
//*  4) Do a global change of MXG.MXG.SOURCLIB to the name of the     *
//*     PDS containing the original MXG source library.               *
//*                                                                   *
//*  5) Do a global change of MXG.MXG.FORMATS to the name             *
//*     of your MXG format library.                                   *
//*                                                                   *
//*  6) Do a global change of YOUR.ITRM to the high-level             *
//*     qualifiers for this application on your z/OS system.          *
//*     Make sure this qualifier has been set correctly for the       *
//*     CONFIG= parameter AND for the SITELIB data library,           *
//*     which contains your site specific settings.                   *
//*                                                                   *
//*  7) Do a change of YOUR.ITRM.PDB to the high-level                *
//*     qualifier for your PDB. This PDB should have one or more      *
//*     tables already in it if your site administrator               *
//*     has included inquire task(s) for the batch utility.           *
//*                                                                   *
//*******************************************************************
//SASCPE EXEC SAS,WORK='10500,2000',
//       CONFIG='your.itrm.CPMISC(CMCONFIG)'
//SMF     DD DSN=your.smf.data,DISP=SHR
//SYSIN   DD DATA,DLM='$$'

  *-------------------------------------------------------------------*
  * To run this application you must first submit %CPSTART.           *
  * %CPSTART allocates this software and the PDB.                     *
  * The MXGLIB= and MXGSRC= parameters are required for the           *
  * MXG software tool used  within %CMPROCES below.                   *
  * The SITELIB= parameter must be specified.                         *
  * The SITEACC= parameter must be set to OLD disposition.            *
  * The ROOTSERV=, SHARE=, and SITESHR= parameters should only        *
  * be used if SAS/SHARE is being used at                             *
  * your site.                                                        *
  * If the _RC= parameter is nonzero from %CPSTART,                   *
  * check the explanatory message in the SAS log.                     *
  *                                                                   *
  * The input stream should begin with a call to the %CPSTART         *
  * macro, with MODE=BATCH specified. Then, the input stream can      *
  * have SAS statements, other calls to SAS IT Resource               *
  * Management macros, etc.                                           *
  *-------------------------------------------------------------------* ;


  /* verify root= pdb= sitelib= mxglib= mxgsrc= in the following */
  %CPSTART(MODE=BATCH,
           ROOT='your.itrm.',
           ROOTSERV=,
```

```
          PDB='your.itrm.pdb.',
          DISP=OLD,
          SHARE=N/A,
          SITELIB='your.itrm.SITELIB',
          SITEACC=OLD,
          SITESHR=,
          MXGLIB=mxg.mxg.formats,
          MXGSRC=('mxg.userid.sourclib' 'mxg.mxg.sourclib'),
          _RC=cpstrc
         );

  %PUT CPSTART return code is &cpstrc;


  *-------------------------------------------------------------*
  * The following %CPBATCH macro runs the list of scheduled    *
  * tasks that have been set up by your site administrator.    *
  * You must code the ENV= parameter and capitalize its value. *
  *-------------------------------------------------------------* ;
  %CPBATCH(ENV=BACK);

$$
//
```

Processing Notes

1. Before you submit the job for the first time, you must perform the following steps, typically from the menu interface of this application:
   - **a** Estimate the space requirements for your PDB and allocate space accordingly. You can accomplish this through the client user interface.
   - **b** Add table definitions to the PDB, as required.
   - **c** Edit the table definitions to adjust durations for retention of detail-, day-, week-, month-, and year-level data, as required.
   - **d** Add, update, or delete variables in the tables, as required.
   - **e** Edit the variable statistics to adjust which statistics are requested for day, week, month, and year levels, as required.
   - **f** Verify that all tables and variables that you want to use are marked KEPT=YES.
2. The %CPSTART macro invokes the SAS IT Resource Management software and allocates the PDB.

   The MODE= parameter must be set to *batch*.

   The ROOT= parameter specifies the high-level qualifiers in the data set names for SAS IT Resource Management software. For example, if the PGMLIB library is located in a SAS library that is installed at your site with the name *your.itrm.pgmlib*, then use *your.itrm* as the value for the ROOT= parameter. Similarly, the PDB= parameter specifies the high-level qualifiers in the data set names for the SAS data libraries in the PDB. For example, if the PDB is in libraries that are all named *your.itrm.pdbname.library*, where *library* varies depending on the library name (such as DETAIL, DAY, WEEK, MONTH, and YEAR), then use *your.itrm.pdbname* as the value for the PDB= parameter.

   The MXGLIB= parameter specifies the fully qualified name of the SAS library that contains the SAS formats that are provided by MXG. The MXGSRC=

parameter specifies fully qualified names of one or more PDSs that contain MXG SOURCLIB source entries and user overrides to these original entries. List the PDSs in search order. If the same member name exists in more than one of the PDSs, then the member that is used is the first one that is encountered. Thus, typically, list the PDS that has your modified source entries in front of the PDS with the MXG source code.

3 The %CPBATCH macro runs the tasks that were set up by using the **Batch Utility** item on the main menu of the SAS IT Resource Management GUI to run on days that match the day of this run.

SAS IT Resource Management Macros Grouped by Task

SAS IT Resource Management provides macros that can be run in batch mode to perform many of the same tasks that can be performed through the user interface. For example, you can use the batch macros to process your data, manage and update your PDB, create report definitions on your data, and generate reports on your data.

For information on using any of the macros during recovery from an error condition, see "Troubleshooting Batch Jobs That Fail" on page 655.

The following list of macros and overviews is divided into groups that are based on the types of tasks that the macros perform:

- "Macros for Building and Managing the PDB" on page 29
- "Macros Used for Analyzing Data" on page 235
- "Using the %CPDDUTL Macro" on page 567 and "%CPDDUTL Macro Control Statements" on page 571.

The Reporting macros can run on either your client system or your server system. However, you can run the other macros only on your server system, because many of the macros require write access to the PDB. For more information, see the section "Actions and Access Rights Required" in the chapter "Setup: Introduction" in the *SAS IT Resource Management User's Guide*.

For more information on SAS IT Resource Management macros, see "Overview of SAS IT Resource Management Macros" on page 1 and "Macro and Syntax Document Conventions" on page 3.

Many of the tasks that you can perform with these report macros can also be performed using the **Reporting** tab on the SAS IT Resource Management client GUI. For more information about the **Reporting** tab, see the section "The Reporting Tab" in the chapter "General: Getting Started" in the *SAS IT Resource Management User's Guide*.

CHAPTER 2

Administration Macros

Macros for Building and Managing the PDB

The following macros enable you to perform tasks such as processing data into the performance database, checking for duplicate data, copying the data dictionary and data, purging data from the data dictionary, and deleting data and other components within the data dictionary.

- %CMEXTDET - processes detail-type data directly into the PDB (z/OS) (see "%CMEXTDET" on page 31).
- %CMFTPSND - sets up control statements and invokes FTP to copy PDS members to any accessible FTP host (z/OS) (see "%CMFTPSND" on page 37).
- %CMPROCES - processes data into the active PDB (z/OS) (see "%CMPROCES" on page 41).
- %CPACCFMT - creates user and group formats for ACCUNX collector (see "%CPACCFMT" on page 52).
- %CPACCUTL - lists/adds/deletes OS/OS release format information for the ACCUNX collector (see "%CPACCUTL" on page 55).
- %CPARCRET - retrieves archived data (see "%CPARCRET" on page 61).
- %CPAVAIL - creates a table and view that can be used for availability reporting (see "%CPAVAIL" on page 64).
- %CPBATCH - runs scheduled work (z/OS) (see "%CPBATCH" on page 71).
- %CPC2RATE - converts counters to rates (see "%CPC2RATE" on page 73).
- %CPCAT - copies control statements to and from SAS catalog source entries (see "%CPCAT" on page 75).
- %CPDBCOPY - copies the contents of one PDB to another PDB (see "%CPDBCOPY" on page 77).
- %CPDBKILL - deletes the data and table definitions from a PDB and initializes the data dictionary (see "%CPDBKILL" on page 79).
- %CPDBPURG - purges data from a PDB (see "%CPDBPURG" on page 80).
- %CPDDUTL - provides for the maintenance of the Data Dictionary Using %CPDDUTL (see "Using the %CPDDUTL Macro" on page 567).
- %CPDD2RPT - generates HP VPPA report parameter file from SAS IT Resource Management tables (see "%CPDD2RPT" on page 82).
- %CPDEACT - deactivates the active PDB (see "%CPDEACT" on page 84).
- %CPDUPCHK - checks for duplicate data when you are processing data into the PDB (see "%CPDUPCHK" on page 86).
- %CPDUPDSN - creates temporary data sets that are used when you are checking for duplicate data (see "%CPDUPDSN" on page 96).
- %CPDUPINT - loads macro definitions that are used by other macros that check for duplicate data (see "%CPDUPINT" on page 98).
- %CPDUPUPD - updates datetime information for macros that check for duplicate data (see "%CPDUPUPD" on page 98).
- %CPEDIT - edits or deletes data in the active PDB (see "%CPEDIT" on page 99).
- %CPEISSUM - summarizes PDB data that will potentially be used to download (see "%CPEISSUM" on page 102).
- %CPENTCPY - creates HTML files from .SOURCE files (see "%CPENTCPY" on page 287).

- %CPFAILIF - terminates a SAS job (with a specified exit code), based on an expression (see "%CPFAILIF" on page 120).
- %CPHDAY - adds, deletes, or modifies site holidays and sets shift code for holidays (see "%CPHDAY" on page 122).
- %CPINSPKG - installs the contents of a SAS IT Resource Management package (see "%CPINSPKG" on page 124).
- %CPLOGRC - redirects the SAS log to a temporary file and scans the file for user-specified text string(s) (UNIX and Windows) (see "%CPLOGRC" on page 128).
- %CPLVLTRM - ages data from a specific level or a specific table (see "%CPLVLTRM" on page 133).
- %CPPDBOPT - sets and copies PDB options (properties) (see "%CPPDBOPT" on page 135).
- %CPPKGCOL - creates a package of user-written collector-support entities (see "%CPPKGCOL" on page 144).
- %CPPROCES - processes data into the active PDB (portable) (see "%CPPROCES" on page 154).
- %CPRAWLST - produces a list of records from a raw data file (see "%CPRAWLST" on page 166).
- %CPREDUCE - reduces data to one or more summary levels (see "%CPREDUCE" on page 167).
- %CPRENAME - renames tables or variables in the active PDB (see "%CPRENAME" on page 170).
- %CPRPT2DD - lists SAS IT Resource Management tables and variable modifications for HP VPPA data (see "%CPRPT2DD" on page 173).
- %CPRPTPKG - reports on the contents of a SAS IT Resource Management package (see "%CPRPTPKG" on page 177).
- %CPSEP - cleans up the name of a path or high-level qualifiers (see "%CPSEP" on page 179).
- %CPSTART - starts SAS IT Resource Management software (see "%CPSTART" on page 181).
- %CPTFORM - builds a view that transforms data (see "%CPTFORM" on page 190).
- %CPUNCVT - converts a PDB from SAS IT Resource Management 2 format to SAS IT Resource Management 1 format (see "%CPUNCVT" on page 196).
- %CSATR2DD - creates data dictionary statements for use with Aprisma SPECTRUM data (see "%CSATR2DD" on page 197).
- %CSCSIFMT - builds site-specific formats for Aprisma SPECTRUM data (see "%CSCSIFMT" on page 202).
- %CSPROCES - processes data into the active PDB (UNIX) (see "%CSPROCES" on page 204).
- %CSSNMSCH - creates user-defined tables for SunNet Manager data (see "%CSSNMSCH" on page 219).
- %CWPROCES - processes data into the active PDB (Windows) (see "%CWPROCES" on page 220).

Many of the tasks that you can perform with these macros can also be performed using the SAS IT Resource Management server GUIs. For more information about using the server GUIs, see

the section "The Administration Tab" in the chapter "General: Getting Started" in the *SAS IT Resource Management User's Guide*.

%CMEXTDET

Processes detail-like data directly into the PDB (z/OS only)

%CMEXTDET Overview

Typically, raw data is processed into the detail level of the PDB and then reduced from the detail level to the summary levels. In some cases, however, the raw data already looks so similar to what would be written to the detail level of the PDB that the process step can be omitted. In these cases, you can use the %CMEXTDET macro to create DATA step views for use by the reduce step. When reduce obtains data through these views, the data appears the way it would appear if the reduce step were getting the data from the detail level.

For example, you can use this macro to enable %CPREDUCE to summarize current detail-like data from a daily library in a production MXG PDB. Additionally, you can use this macro to enable %CPREDUCE to summarize current detail-like data from a library that has a week's worth of data that has been appended from daily libraries in a production MXG PDB. "Current" means that you are summarizing data that is newer than the data that is already in the SAS IT Resource Management PDB's summary levels.

You can also use this macro to enable %CPREDUCE to summarize old data. Bringing old data into a SAS IT Resource Management PDB is also called backloading data. "Backloading" means that you are summarizing data that is older than at least some of the data that is already summarized into the SAS IT Resource Management PDB.

In the following description, the term *external library* refers to any of these libraries that contain new or old detail-like data from an MXG production PDB. The external library can be on either DASD or tape.

%CMEXTDET Syntax

%CMEXTDET(
<*tablist*>
,<*function*>
,**EXTLIB=***libref*
<,**BACKLOAD=**GETDATE | *date*>
<,**BACKFRCE=**YES | NO>
<,**DURATION=***value*>
<,**LAND13=**YES | NO>
<,**NETMASTR=**YES | NO>
<,**_RC=***macro-var-name*>
<,**VIEWLIB=***libref*>);

Details

tablist

lists the tables for which you want views to be built. This parameter is not required. If you omit this parameter, then the external library that is specified by the EXTLIB= parameter is searched and views are built for all SAS data sets that correspond to tables in the SAS IT Resource Management master data dictionary. If the active PDB's data dictionary does not have all those tables, then any table definitions that are missing are added from the master data dictionary.

function
specifies whether existing views are to be rebuilt (and overwritten). Use the function parameter only if you have modified the table or variable definitions in the active PDB or if you have modified SAS data set or column definitions in the external library.

If the function parameter is not specified, then existing views remain unchanged. *This is the default.*

If the function parameter is specified, then the value must be **`setup`**, and existing views are rebuilt (and overwritten). If you specify this parameter and you specify a list of tables (using the *tablist* parameter), then views are rebuilt for the specified tables only. If you do not also specify a list of tables, then views are rebuilt for all table/data set matches.

EXTLIB=*libref*
specifies the SAS libref or DDname of the external library that contains the detail-like data that you want to summarize into the active SAS IT Resource Management PDB. The data can be newer than or older than the data that is already in the SAS IT Resource Management PDB. *This parameter is required.*

The external library can reside on tape or DASD. You can allocate the library outside SAS by using a DD statement. Or you can allocate the library within the SAS and SAS IT Resource Management code by using a SAS LIBNAME statement that is specified in the input jobstream that precedes the call to %CMEXTDET. (You might prefer to allocate the library outside SAS if the library resides on tape.)

When the %CPREDUCE macro reads the data, it uses not this libref but the libref that is specified by the VIEWLIB= parameter, because %CPREDUCE needs to use the views that are created by %CMEXTDET's VIEWLIB= parameter to find the data residing in the library that is specified in this parameter.

BACKLOAD=GETDATE | *date*
when you are backloading data, this parameter is used to read the date on the external library or to identify the date when the external data was created. Specify this parameter only if the data in the external library is older than at least some of the data that has already been summarized into the SAS IT Resource Management PDB (that is, if you are backloading data). *If you are backloading data, then this parameter is required.*

If the external library has not been copied since it was created in relation to the MXG production PDB, then the date on the external library correctly indicates the date that the data was last written to the external library. In that case, you can specify BACKLOAD=GETDATE, which causes %CMEXTDET to read that date and check it against its list of dates of external libraries whose data was previously backloaded. If the date is not already on the list, then %CMEXTDET adds the date to the list and continues. If the date is already on the list, then %CMEXTDET stops so that the apparently duplicate data cannot be summarized into the SAS IT Resource Management PDB. (If you have a reason to summarize old data from more than one external library with the same external date, then see the BACKFRCE= parameter.)

If you have copied the data in the original external library to a new data set, then the new data set's date is probably different from the date on the original data set. If all the other data sets whose data you are backloading are undergoing duplicate checking by using their original dates, then use the BACKLOAD=date setting to specify the date of the original data set. For the date value, use the format *ddmmmyyyy*, as in 08JUN2003. If you do not know the exact date, then enter the approximate date that the original data set was created.

For additional information, see the BACKFRCE= parameter.

BACKFRCE=YES | NO

indicates whether an external library's old data should be summarized if old data was already summarized from an external library with the same date.

Use the BACKFRCE= parameter only if you have specified the BACKLOAD= parameter. BACKFRCE=YES indicates that even if data was previously backloaded from an external library with the same date as this external library, then data from this external library should also be allowed to be summarized into the SAS IT Resource Management PDB. In effect, this setting disables duplicate checking for backloaded data. You might need to use this setting if you are summarizing old data from several external libraries that have the same external date. For example, one external library might contain saved daily SMF data from an MXG production PDB, and another might contain saved daily DB2 data from an MXG production PDB.

BACKFRCE=NO indicates that if old data was summarized from another external library with the same date as this external library, then data from this external library is not to be summarized. In effect, this setting enables duplicate checking for backloaded data. *The default is BACKFRCE=NO.*

Note that the date that %CMEXTDET uses is the external date on the external library. The dates on the observations that are within the data set might or might not be the same. (The %CPREDUCE macro ignores the external date on the external library and uses the dates on the detail-like observations within the external library, in exactly the same way that it uses dates on observations that come from the detail level.)

For additional information, see the BACKLOAD= parameter.

DURATION=*value*
identifies the number of seconds that should be used for the DURATION variable for interval tables, when the incoming data does not have a duration variable. *The default value is 3600, or one hour.* If the incoming data has a DURATION variable and the values of DURATION are not missing, then this parameter is ignored.

LAND13=YES | NO
specifies whether or not you have Landmark's Monitor for CICS, Version 1.3 (or later), installed. Specify this parameter only if you are summarizing CICS monitor data. *The default value is NO.* The tables and variables are different for each version of that software.

NETMASTR=YES | NO
specifies whether or not you have the Netmaster product installed. *The default value is NO.*

Use this parameter only if you process TYPE39 subtype 6 SMF records. When Netmaster is installed, the incoming observations contain interval data. When Netmaster is not installed, the incoming observations contain event data.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the

macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

VIEWLIB=*libref*

specifies the libref of the library where the DATA step views are to be written. *The default (and recommended) value is COLLECT*, which is the libref of the COLLECT library in the active PDB. If you do not use COLLECT, then another recommended libref is ADMIN, which is the libref of the ADMIN library in the active PDB. The librefs COLLECT and ADMIN are defined automatically by the %CPSTART macro.

The views in the library whose libref is specified by the VIEWLIB= parameter point to the data that is to be read from the library whose libref is specified by the EXTLIB= parameter.

When the %CPREDUCE macro reads that data, it uses the libref that is specified by its DETAIL= parameter to locate the views. Thus, the libref that you specify for the %CPREDUCE macro's DETAIL= parameter is typically the same as the libref that you specify for %CMEXTDET macro's VIEWLIB= parameter.

%CMEXTDET Notes

This macro is available only on z/OS. Before you use this macro, read the implementation instructions that are available online in the Help Index in the SAS IT Resource Management GUI for z/OS. To access the Help Index, follow this path from the main menu:

Help ▶ Help Index ▶ %CMEXTDET reducing external detail data ▶ ItemActions ▶ Browse Help

In the %CMEXTDET macro, you can use the EXTLIB= parameter to specify a pointer to the external library, and you can use the VIEWLIB= parameter to specify where the views are to be created. In the %CPREDUCE macro, use the DETAIL= parameter to specify where the views were created. The views point to the SAS data sets in the external library.

Each SAS IT Resource Management table has an external name. (External name is one of the properties of a table.) Each SAS data set in the external library has a name. A data set in the external library corresponds to a table if the data set's name matches the table's external name. In the case of a match, a view is created to enable the table to obtain data from the data set. The view is given the same name as the data set and the table's external name.

For all matches, if the table's definition is not already in the PDB's data dictionary, then %CMEXTDET adds the table definition to the PDB from the master data dictionary.

You can use SAS IT Resource Management to view and report on the data in the summary levels of the SAS IT Resource Management PDB (the day, week, month, and year levels). Use MXG to view and report on the detail-like data in the external library.

%CMEXTDET Examples

Note: Many of the following examples contain a call to the %CPREDUCE macro in which no table list is specified; therefore, the detail-like data is to be obtained from COLLECT's library (which contains the views that obtain the data from MXGWEEK's library). The %CPREDUCE macro summarizes the detail-like data for each table

within the PDB's data dictionary that has the following characteristics: the table's Kept status is set to Yes; the table has at least one summary level with a nonzero age limit; the table has at least one variable with Kept status set to Yes; and COLLECT's library has a view that corresponds to the table. △

Example 1

This example assumes that you are reducing data from an MXG weekly backup tape and that the data is current. It also assumes that you have already defined the DDname (or libref) MXGWEEK to point to the external library on the tape.

In the call to the %CMEXTDET macro, the EXTLIB= parameter identifies MXGWEEK's library as the location of the detail-like data that is to be summarized.

The table list parameter is not specified, the FUNCTION parameter is not specified, and the VIEWLIB= parameter defaults to the value COLLECT. Thus, if the views do not already exist in COLLECT's library, the %CMEXTDET macro creates views in COLLECT's library for all tables that correspond to SAS data sets in MXGWEEK's library. The macro also adds table definitions to the PDB for any of these tables that are not already defined in the PDB.

```
%cmextdet(,extlib=mxgweek);
%cpreduce(detail=collect);
```

Example 2

This example assumes that you are reducing data from an MXG weekly backup tape and that the data is current. It also assumes that you have defined the DDname (or libref) MXGWEEK to point to the data set on the tape.

In the call to the %CMEXTDET macro, the EXTLIB= parameter identifies MXGWEEK's library as the location of the detail-like data that is to be summarized.

The table list parameter is not specified, a value (SETUP) is specified for the function parameter, and the VIEWLIB= parameter defaults to the value COLLECT. Thus, even if the views already exist in COLLECT's library, the macro re-creates views in COLLECT's library for all table/data set matches. If views already exist, then they are overwritten. The macro also adds table definitions to the PDB for any of these tables that are not already defined in the PDB.

```
%cmextdet(,setup,extlib=mxgweek);
%cpreduce(detail=collect);
```

Example 3

This example assumes that you are reducing data from an MXG weekly backup tape and that the data is current. It also assumes that you have defined the DDname (or libref) MXGWEEK to point to the external library on the tape.

In the call to the %CMEXTDET macro, the EXTLIB= parameter identifies MXGWEEK's library as the location of the detail-like data that is to be summarized.

The table list parameter is specified (XTY70), the FUNCTION parameter is not specified, and the VIEWLIB= parameter defaults to COLLECT. Thus, if a view does not already exist in COLLECT's library for the XTY70 table, then the view is created in COLLECT's library. If the XTY70 table is not already defined in the PDB, then the macro adds its definition to the PDB.

In the call to the %CPREDUCE macro, the value (XTY70) of the table list parameter is specified, and the data is to be obtained from COLLECT's library (which contains the view that obtains the data from MXGWEEK's library). The %CPREDUCE macro summarizes the detail-like data for the XTY70 table only (regardless of the Kept status of the table) and uses the view in COLLECT's library to obtain the data.

```
%cmextdet(xty70,extlib=mxgweek);
%cpreduce(xty70,detail=collect);
```

Example 4

This example assumes that you are reducing data from an MXG weekly backup tape and that the data is current. It also assumes that you have defined the DDname (or libref) MXGWEEK to point to the external library on the tape.

In the call to the %CMEXTDET macro, the EXTLIB= parameter identifies MXGWEEK's library as the location of the detail-like data that is to be summarized.

The table list parameter is specified (XTY70), a value (SETUP) is specified for the function parameter, and the VIEWLIB= parameter is specified (ADMIN). Even if ADMIN's library already has a view for the XTY70 table, the view is created in ADMIN's library (and overwrites the previous view for that table, if there was one). If the XTY70 table is not already defined in the PDB, then the macro adds its definition to the PDB.

```
%cmextdet(xty70,setup,extlib=mxgweek,viewlib=admin);
%cpreduce(detail=admin);
```

Example 5

This example assumes that you are reducing data from an MXG weekly backup tape and that the data is older than at least some of the data already in the SAS IT Resource Management PDB. It also assumes that you have defined the DDname (or libref) MXGWEEK to point to the external library on the tape.

In the call to the %CMEXTDET macro, the EXTLIB= parameter identifies MXGWEEK's library as the location of the detail-like data that is to be summarized.

The table list parameter is not specified, the function parameter is not specified, and the VIEWLIB= parameter defaults to the value COLLECT. Thus, if the views do not already exist in COLLECT's library, then the %CMEXTDET macro creates views in COLLECT's library for all table/data set matches. The macro also adds table definitions to the PDB for any of these tables that are not already defined in the PDB.

Because the BACKLOAD= parameter is set to GETDATE, the date on the tape's external library is checked against the %CMEXTDET macro's list of the dates on already summarized external libraries. If the date is on the list, then the %CMEXTDET macro aborts both SAS IT Resource Management and SAS (because BACKFRCE=YES is not specified). If the date is not on the list, then %CMEXTDET adds the date to the list and continues.

```
%cmextdet(extlib=mxgweek,backload=getdate);
%cpreduce(detail=collect);
```

Example 6

This example assumes that you are reducing data from two MXG daily external libraries and that the data is current. It also assumes that you have defined the DDname (or libref) MXGSMF to point to one external library and the DDname (or libref) MXGDB2 to point to the other external library. Each table for which data is to be summarized is represented by a SAS data set in one of the libraries (but not both libraries).

In the calls to the %CMEXTDET macro, the EXTLIB= parameters identify MXGSMF's library and MXGDB2's library as the locations of the detail-like data (SMF data and DB2 data) that is to be summarized.

The table list parameter is not specified, the function parameter is not specified, and the VIEWLIB= parameter defaults to the value COLLECT. Thus, if the views do not already exist in COLLECT's library, then the %CMEXTDET macro creates views in

COLLECT's library for all table/data set matches. The macro also adds table definitions to the PDB for any of these tables that are not already defined in the PDB.

```
%cmextdet(extlib=mxgsmf);
%cmextdet(extlib=mxgdb2);
%cpreduce(detail=collect);
```

Example 7

This example is similar to Example 6, but in this example it is possible that there are tables whose data set names are common to (identical in) both the external libraries. (One library might have data from one set of machines and the other library might have data from another set of machines.) Each table for which data is to be summarized is represented by a SAS data set in one library, the other library, or both libraries.

In this case, you need two storage locations for the views. The tables that correspond to SAS data sets that are in both MXG libraries need to have one set of views that point to the SAS data sets in MXG1's library and another set of views that point to the SAS data sets in MXG2's library. (If the views were stored in the same place, then the second set of identically named views would write over the first set.)

```
%cmextdet(extlib=mxg1,viewlib=collect);
%cmextdet(extlib=mxg2,viewlib=admin);
%cpreduce(detail=collect);
%cpreduce(detail=admin);
```

Example 8

This example is similar to Example 7, but in this example suppose that the data in the pair of external libraries is being backloaded. That is, the data in both external libraries is from a date that is earlier than at least some of the data that was already summarized into the SAS IT Resource Management PDB.

Because the second external library has a date that is the same as the date of the first external library, you must set BACKFRCE=YES in order to force the second external library past %CMEXTDET's duplicate-prevention mechanism.

```
%cmextdet(extlib=mxg1,viewlib=collect,backload=getdate);
%cmextdet(extlib=mxg2,viewlib=admin,backload=getdate,backfrce=yes);
%cpreduce(detail=collect);
%cpreduce(detail=admin);
```

%CMFTPSND

Creates control statements and invokes FTP to transfer PDS members to an accessible FTP host (z/OS only)

%CMFTPSND Overview

The %CMFTPSND macro creates control statements and invokes File Transfer Protocol (FTP) to transfer PDS members to any accessible FTP host. This macro uses FTP to transfer Web output files (generated by SAS IT Resource Management to a PDS on z/OS if the z/OS UNIX File System area is not used) to a directory-based file system (UNIX, Windows, or the z/OS UNIX File System). The %CMFTPSND macro can transfer the "tree structure" if it is produced in PDS form by %CPHTREE and also the Web output from SAS IT Resource Management report macros.

%CMFTPSND Syntax

%CMFTPSND (
dsnlist
,**FTPHOST=***FTP-hostname*
,**FTPUSER=***user-id*
,**FTPUSERP=***user-password*
,**RDIR=***pathname*
<,**CNTLDISP=***disposition*>
<,**CNTLPFX=***text-string*>
<,**DASDUNIT=***unit-name*>
<,**FTPPARM=***text string*>
<,**FTPUCMDS=***text-string*>
<,**MAXDSNS=***number-of–DSNs*>
<,**PROC=***program-name*>
<,**_RC=***macro-var-name*>);

Details

dsnlist
lists the DSNs for all the PDSs that are to be transferred. If a "super" tree is to be transferred, then the list includes the "super" PDS (the PDS that is specified by the SUPERLOC= parameter in %CPHTREE) and includes the PDS whose name ends with the WEB qualifier.

For each single tree within the "super" tree, the list also includes the single tree's top PDS (the PDS that is specified by the SUBLOC= parameter in %CPHTREE) and the single tree's level 0, 1, 2, etc. PDSs.

If PDSs that are unrelated to a "super" tree are to be transferred (that is, PDSs that are created by the reporting macros when OUTPUT=WEB but without the tree structure that is created by %CPHTREE), then the list includes the name of each of the PDSs. If more than one DSN is on the same line, then you must separate the DSNs with one or more blanks.

This parameter is required. For information about the maximum number of DSNs, see the MAXDSNS= parameter.

FTPHOST=*FTP-hostname*
specifies the name of the host to which the files are to be transferred (the *to*-host). FTP server software must reside on this host. *This is a required parameter.*

FTPUSER=*userid*
specifies a userid that is valid on the host that is specified by the FTPHOST= parameter. Typically, this userid is the word "anonymous." If your site does not allow anonymous FTP, then the userid must be a valid userid on the host that is specified by the FTPHOST= parameter. *This is a required parameter.*

FTPUSERP=*user-password*
specifies the password for the userid that is specified by the FTPUSER= parameter. If the userid is "anonymous," then this password is typically your e-mail address, such as **`your-userid@your-organization.com`**. *This is a required parameter.*

RDIR=*pathname*
specifies the full path and name of a directory. The directory will be created if it does not already exist.

If you are transferring a "super" tree, then this directory corresponds to the PDS that was specified by the SUPERLOC= parameter in the calls to

%CPHTREE. Typically, the directory name that is specified is the same as the right-most qualifier of the name of the "super" PDS. The appropriate subdirectories and files will be created (if necessary) and will be written to a subdirectory under this directory.

If PDSs that are unrelated to a "super" tree are to be transferred (that is, PDSs that are created by the reporting macros when OUTPUT=WEB), then this parameter specifies a "grouping" directory under which the appropriate subdirectories and files will be created (if necessary) and written to. It is appropriate to use an arbitrary name for this "grouping" directory. Note that the userid that is specified in the FTPUSER= command must have the authority to create and write to these directories. *This parameter is required.*

CNTLDISP=*disposition*
specifies the disposition of two intermediate z/OS data sets: the data set where the FTP script is built, and the data set that contains the messages that result from running the FTP script. The value of the parameter should be one that you could specify at your site for the DISP parameter on a JCL DD card or in a TSO ALLOCATE command. This is an optional parameter. *The default is CNTLDISP=(NEW, DELETE)*.

For the names of these intermediate z/OS data sets, see the CNTLPFX= parameter. For the unit of these intermediate z/OS data sets, see the DASDUNIT= parameter.

CNTLPFX=*text-string*
The %CMFTPSND macro creates two intermediate z/OS data sets: the data set where the FTP script is built, and the data set that contains the messages that result from running the FTP script. This parameter affects the names of the data sets.

If the value of the CNTLDISP= parameter is not (NEW, DELETE), then the CNTLPFX= parameter specifies the prefix (high-level qualifiers and trailing period) of these intermediate z/OS data sets. The data sets are named &CNTLPFX.FTP.INPUT and &CNTLPFX.FTP.OUTPUT. If the value of the CNTLDISP= parameter is (NEW,DELETE), then the CNTLPFX= parameter is ignored. The intermediate z/OS data sets are named &&FTPIN and &&FTPOUT.

This parameter is not required. *The default prefix is the name of the active PDB (including the trailing period)*.

The ID under which this macro executes on z/OS must have authority to create new data sets with the specified (or default) prefix. For the disposition of these intermediate z/OS data sets, see the CNTLDISP= parameter. For the unit of these intermediate z/OS data sets, see the DASDUNIT= parameter.

DASDUNIT=*unit-name*
specifies the unit name or keyword of two intermediate z/OS data sets: the data set where the FTP script is built, and the data set that contains the messages that result from running the FTP script. The value of the parameter should be one that you can specify at your site for the UNIT= parameter on a JCL DD card or a TSO ALLOCATE command.

This is an optional parameter. *The default is DASDUNIT=SYSDA*.

For the disposition of these intermediate z/OS data sets, see the CNTLDISP= parameter. For the names of these intermediate z/OS data sets, see the CNTLPFX= parameter.

FTPPARM=*text-string*
specifies a text string that contains any parameters that you would normally pass to the FTP program on the *from*-host if you called the program directly. For example, to pass the Austrian/German translate table to the FTP program on the z/OS host, specify

```
FTPPARM=-T
AUSGER
```

FTPUCMDS=*text-string*
specifies FTP commands that should be executed on the *to*-host (the host that is specified by the FTPHOST= parameter). The commands are executed

- after the *from*-host establishes communications with the *to*-host and the session is authenticated (using the values specified by the FTPUSER= parameter and the FTPUSERP= parameter)
- before the change to (and creation of, if necessary) the "super" directory (which is specified by the RDIR= parameter).

For example, to make **Read** the default permission for your entire group, you could specify

```
FTPUCMDS=umask 022
```

To specify multiple FTP commands, use the exclamation point (!) to separate each command from its adjacent command(s).

This parameter is optional. The default is that no additional commands will be executed.

MAXDSNS=*number-of-DSNs*
specifies the maximum number of DSNs in the *dsnlist* parameter. This parameter is optional. *The default is MAXDSNS=256*.

PROC=*program-name*
specifies the name of the program that is to be used on the z/OS host (the *from*-host) to transfer the files. This parameter is optional. *The default is PROC=FTP*.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CMFTPSND Notes

CAUTION:

Do not delete or edit the FTPCNTL or TCPSEND members. The %CMFTPSND macro depends on those members' being in the state in which other SAS IT Resource Management macros left them. It is very important that you check the SAS log and

the z/OS data set that contains the messages that are generated while running FTP (for more about this data set, see the CNTLPFX=, CNTLDISP=, and DASDUNIT= parameters). △

CAUTION:

Because the FTP program that this macro runs is not a SAS IT Resource Management program, the meaning of the return code is different from the meaning of the return code for other SAS IT Resource Management macros. For example, a value of zero for the return code does not necessarily indicate successful completion. △

%CMFTPSND Example

This example transfers files from two partitioned data sets.

```
%CMFTPSND(  USERID.ITRM.WEB.OUTPUT1
            USERID.ITRM.WEB.OUTPUT2,
            ftphost=ftpserv1.ourcorp.com,
            ftpuser=sasusr,
            ftpuserp=sasusrp,
            rdir=\\server1\itsv\reports\  );
```

For additional examples of using this macro to transfer files, see the examples for the %CPHTREE macro (see "%CPHTREE" on page 318). Examples 5 and 6 illustrate the use of %CMFTPSND after a call to a reporting macro.

Also, if you use the SAS IT Resource Management QuickStart Wizard and specify an area that is not used for the z/OS UNIX File System as the location of the report structure, you can see an example of a call to %CMFTPSND in the xFTPHTML job that the QuickStart Wizard places in the .QS.CNTL PDS.

%CMPROCES

Processes data into the active PDB (z/OS only)

%CMPROCES Overview

The %CMPROCES macro reads, validates, and transforms performance data into the detail level of the active PDB from the files that contain the logged data. In addition, %CMPROCES updates data dictionary entries that describe the amount of data in the PDB and the most recent date and time of processing data into the detail level of the tables.

%CMPROCES Syntax

%CMPROCES(
<*rawdata*>
,<*tablist*>
,**COLLECTR=***collector-name*
,**TOOLNM=***tool-name*
<,**DUPMODE=**INACTIVE | FORCE | DISCARD | TERMINATE>
<,**EXITSRC=***exit-source*>
<,**GENLIB=***libref*>
<,**JES=**JES2 | JES3>

<,**_RC=***macro-var-name*>
<,**UNIT=**DISK | TAPE | CART | other>
<,**VOLSER=***volser*>
<,**WKCMPRES=**YES | NO>);

Details

rawdata
is the complete name of the data set that contains the logged data that is to be processed. This positional parameter is not required. If it is specified, then it must be specified in the first position on this macro. If you omit this parameter, then use a comma as a placeholder.

The name of the data set can be any of the following:

- a Generation Data Group (GDG), in which case the generation number is 0 for current, -1 for previous, -2 for one before that, and so on.
- any other collection of SMF-style data.

If you specify COLLECTR=SMF, then your options are as follows:

- Omit the *rawdata* parameter and specify in your JCL a DD name of SMF that points to the SMF data set. This is the preferred method.
- Specify the name of the SMF data set as the *rawdata* parameter.
- Omit the *rawdata* parameter and use the SAS FILENAME statement to assign a libref of SMF to specify the SMF data set.

If you define the data set with a DD statement or a SAS FILENAME statement, then you must specify the complete fileref for the file. (See the following table for a list of filerefs for the different collectors.)

Table 2.1 Filerefs for Each Collector

Collector	Fileref
DCOLLECT	DCOLLECT
EREP	EREP
IMF	IMSLOG
NTSMF	NTSMF
SMF	SMF
TMONCIC8	MONICICS
TMONCICS	MONICICS
TMONDB2	TMDBIN
TMON2CIC	MONICICS
TMS	TMC
TPF	TPFIN
VMMON	MWINPUT
VMMONV	VMINPUT

If you specify COLLECTR=GENERIC, then you must not specify the *rawdata* parameter. Instead, you must specify the GENLIB= parameter.

tablist

lists the names of the tables into which the logged data should be processed. *This positional parameter is not required*. If you do not specify this parameter, then use a comma as a placeholder. If you specify more than one table, then use blanks to separate the tables in the list.

If you specify one or more tables, then the data in these tables is processed into the PDB.

If you do not specify a list of tables, then the data is processed into the PDB for all tables that meet the following criteria:

- tables that are associated with the collector that is specified in the COLLECTR= parameter
- tables that have a status of KEPT=Y
- tables that have at least one variable at detail level with a KEPT status of YES.

If you specify a table in the list and the table is not found in the active PDB, then the table definition that is supplied is added to the active PDB from the master data dictionary.

COLLECTR=*collector-name*
specifies the name of the data collector or data source that provides the raw data. The value that you specify for the COLLECTR= parameter depends on which process macro you use. The following example uses %CSPROCES with the SUNETMGR collector:

```
%CSPROCES( ...collectr=sunetmgr,...);
```

Also, the TOOLNM= parameter may be required depending on which value you specify for the COLLECTR= parameter. The following example uses %CWPROCES with the WEBLOG collector:

```
%CWPROCES( ...collectr=weblog, toolnm=clf,...);
```

See the following COLLECTR - TOOLNM Parameter Values table for valid values of the COLLECTR= parameter and the TOOLNM= parameter. In the COLLECTR= Value and TOOLNM= Value columns of this table, the following process macro abbreviations apply:

%CM for %CMPROCES

%CP for %CPPROCES

%CS for %CSPROCES

%CW for %CWPROCES.

Table 2.2 COLLECTR - TOOLNM Parameter Values

COLLECTR= Value	Collector Description	TOOLNM= Value
GENERIC for %CM, %CP, %CS, and %CW	data collector or data source for which SAS IT Resource Management does not supply table definitions and the user has not installed table definitions	SASDS for %CM SASDS or CHARDELIM for %CP, %CS, and %CW
ACCUNX for %CP	UNIX accounting	SASDS

COLLECTR= Value	Collector Description	TOOLNM= Value
DCOLLECT for %CM	IBM DFSMS Data Collection Facility	MXG
EREP for %CM	IBM SYS1.LOGREC Error Recording	MXG
ETEWATCH for %CP	End-To-End Watch (transaction response measurement) by Candle	ETE12 (ETEWatch 1.2 from Candle) ETE13 (ETEWatch 1.3 from Candle) EBA (EBA logs from Candle)
HP-MWA for %CS and %CW	HP MeasureWare	MWA-BE or MWA-LE (See Note 6 following the table)
HP-OV for %CS and %CW	IBM Tivoli NetView or HP Network Node Manager	not applicable
HP-OV-C for %CS	IBM Tivoli NetView or HP Network Node Manager + Coerce	not applicable
HP-OVPA for %CS and %CW	HP OpenView Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
HP-PCS for %CS and %CW	HP Performance Collection Software	MWA-BE or MWA-LE (See Note 6 following the table)
HP-PCS for %CP	HP OpenView Reporter	SASACCESS
HP-VPPA for %CS and %CW	HP VantagePoint Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
IMF for %CM	Boole & Babbage IMF from BMC	MXG
LSPW for %CS and %CW	Landmark Performance Works	CHARDELIM

COLLECTR= Value	Collector Description	TOOLNM= Value
NETCONV for %CP	Cisco IOS NetFlow	NETFLOW
NTSMF for %CM and %CP	Windows with Demand Technology's NTSMF product	MXG for %CM SASDS for %CP
PATROL for %CP	BMC Patrol	SASDS
PROBEX for %CS	Probe/X by Landmark in SunNetMgr format	not applicable
ROLMPBX for %CP	Rolm PBX	SASDS
SAPR3 for %CP	SAP R/3	SASADAPT (SAS IT Management Adapter for SAP) STATBIN (Statistics File) STATRFC (Using SAP R/3 API) (See Note 7 following the table)
SAR for %CP	UNIX sar command (system activity reporting)	SASDS
SITESCOP for %CP	SiteScope by Mercury Interactive	SASDS
SMF for %CM and %CP	IBM z/OS System Management Facilities	MXG
SPECTRUM for %CS	SPECTRUM by Aprisma	not applicable
SUNETMGR for %CS	SunNet Manager and Enterprise Manager	not applicable
TMON2CIC for %CM	Landmark "The Monitor for CICS/ESA 2"	MXG

COLLECTR= Value	Collector Description	TOOLNM= Value
TMONCIC8 for %CM	Landmark "The Monitor for CICS" (older)	MXG
TMONCICS for %CM	Landmark "The Monitor for CICS" V 1.3 and later	MXG
TMONDB2 for %CM	Landmark "The Monitor for DB2"	MXG
TMS for %CM	CA-TMS (Release 5 or later)	MXG
TPF for %CM	IBM Transaction Processing Facility	MXG
TRAKKER for %CS	Trakker by Concord Communications	not applicable
VISULIZR for %CP	Visualizer by BMC Software	SASACCESS
VMMON for %CM	VM Monitor	MXG
WEBLOG for %CP	Web server log	CLF (Common Log Format) ELF (Extended Log Format) IISORIG (Microsoft IIS original Weblog format) WEBETE (ETEWatch Web browser logs from Candle) WEBEBA (EBA logs from Candle)

1 *For COLLECTR=GENERIC:* For more information about using the Generic Collector Facility, see the chapter "Setup Case 3" in the *SAS IT Resource Management User's Guide* and the chapter "Setup Case 4" in the *SAS IT Resource Management User's Guide*, and the chapter "Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

   For more information about installing user-written or consultant-written table definitions, see "%CPPKGCOL" on page 144, "%CPRPTPKG" on page 177, and "%CPINSPKG" on page 124.

2 *For %CMPROCES:* The COLLECTR= parameter is required. The TOOLNM= parameter is also required.

3 *For %CPPROCES:* The COLLECTR= parameter is optional. If the parameter is not specified, then the default is COLLECTR=GENERIC. The TOOLNM= parameter is also optional. If the parameter is not specified, then the default is TOOLNM=SASDS.

4 *For %CSPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

5 *For %CWPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

6 *For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA:* The TOOLNM= parameter is used only when you use a PIPE command for the raw data.

For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA, the TOOLNM= value is used to identify the endian type of the data. For example, data from Windows platforms is little-endian and data from most UNIX platforms is big-endian.

Specify TOOLNM=MWA-BE if the incoming data is big-endian (BE). Specify TOOLNM=MWA-LE if the incoming data is little-endian (LE).

If you use a PIPE command for the raw data and you do not specify the TOOLNM= parameter, SAS IT Resource Management attempts to determine the endian type of the incoming data. If the attempt is successful, the processing step continues. If the attempt is not successful, the processing step terminates with a message.

Note: HP OpenView Performance Agent (HP-OVPA) is the latest name for these products: HP-PCS, HP-MWA, and HP-VPPA. △

7 *SAP:*

- *For SAP Release 4.6D and earlier, use TOOLNM=STATBIN or TOOLNM=STATRFC.*

  You must set one macro variable before you run %CPPROCES. Additionally, there are four other optional variables that you might need to set, depending on the data that you are processing.

  The required macro variable is

  - SAPVER - the SAP R/3 version.

  The optional macro variables are

  - SAPSYSNR - the SAP R/3 system number
  - SAPSYSNM - the SAP R/3 system name
  - SAPHOST - the SAP R/3 application server name
  - SAPPLAT - the platform on which the SAP STAT file was generated.

  You need to specify the SAPPLAT macro variable only if you are processing SAP STAT data that was generated in a Windows operating environment. The default value for the SAPPLAT macro variable enables you to read data that is generated in both UNIX and z/OS operating environments. However, if you read data that was generated in a Windows operating environment, then set SAPPLAT to a value of WIN. (For more information, see the examples for the %CPPROCES macro.)

  You must specify the SAPSYSNM, SAPHOST, and SAPSYSNR macro variables only if you use the SAP R/3 collector and only if you process a single data file.

  To populate these macro variables automatically, see the instructions in the RAWDATA= parameter for the %CPPROCES macro.

- *For SAP releases later than 4.6D, use TOOLNM=SASADAPT.*

  You can also use TOOLNM=SASADAPT with some earlier releases of SAP.

8 *If you are working with a Candle collector:*

- Use COLLECTR=ETEWATCH, TOOLNM=ETE12 or ETE13 or EBA when application and transaction analysis is desired.
- Use COLLECTR=WEBLOG, TOOLNM=WEBETE or WEBEBA when processing WebBrowser behavior module data or EBA data for which a hierarchical page analysis (for example, jobs —> Cary —> R&D) is desired.

9 *For COLLECTR=SMF on UNIX or Windows:* See "Example 8: Processing SMF data on UNIX or Windows" in the %CPPROCES Examples.

10 Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

TOOLNM=*tool-name*
typically specifies the name of the software that will be used to transform the raw data and to stage the data into SAS data sets or data set views. The value that is specified for this parameter is based on the value that you specify for the COLLECTR= parameter.

For more information about values of the TOOLNM= parameter, see the COLLECTR= parameter. The COLLECTR= parameter contains a table of COLLECTR= and TOOLNM= combinations and also contains collector-specific and macro-specific notes.

Note: Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*. △

DUPMODE=INACTIVE | FORCE | DISCARD | TERMINATE
specifies how duplicate-data checking is to be handled. *The default value is INACTIVE*.

If you *do not* want to use duplicate-data checking, set DUPMODE=INACTIVE (and, if COLLECTR=GENERIC, check that the staging code does not call the duplicate-data-checking macros).

INACTIVE
does not review the incoming data. Therefore, it allows duplicate data, if there is any, to be processed into the active PDB.

If you *do* want to use duplicate-data checking, set DUPMODE=FORCE, DISCARD, or TERMINATE. To check for duplicate data, the timestamp information for incoming data is compared to the timestamp information for existing data, and the timestamp information for new, non-duplicate data is saved for future data checking.

FORCE
reviews the incoming data and records information on the duplicate data, but still allows (forces) duplicate data to be processed into the active PDB. The

data checking report that is printed in the SAS log provides information about the data that you processed. This value (FORCE) can be used to load data into a table that was not used when processing was done earlier, to load data into a table that was accidentally purged or deleted, and so on.

%CPPROCES and %CMPROCES check whether FORCE is requested by either DUPMODE=FORCE (on the %CPPROCES or %CMPROCES call) or FORCE=YES on the %CPDUPCHK call or both. If either or both are true, the FORCE option in processing is in effect.

Remember to change the DUPMODE= and/or FORCE= settings to their earlier values after you finish using them to allow "duplicate" data into the PDB.

DISCARD
reviews the incoming data and rejects (discards) duplicate data, but allows processing to continue. The data checking report that is printed in the SAS log provides information on the data that was processed and/or discarded.

When DUPMODE=DISCARD, only the FORCE= and TERM= parameters on %CPDUPCHK can affect the FORCE and TERM options in processing.

TERMINATE
reviews the incoming data and terminates the macro if duplicate data is encountered.

%CPPROCES and %CMPROCES check whether TERMINATE is requested by either DUPMODE=TERMINATE (on the %CPPROCES or %CMPROCES call) or TERM=YES on the %CPDUPCHK call or both. If either or both are true, the TERMINATE option in processing is in effect.

In interactive mode, if the TERMINATE option in processing is in effect and duplicate data is detected, then

- the process step terminates
- an ERROR message displays at the end of the SAS log
- the PDB is not deactivated
- the SAS IT Resource Management session continues
- the underlying SAS session continues.

In batch mode, if the TERMINATE option in processing is in effect and duplicate data is detected, the job abends.

Before you can enable duplicate-data checking by means of the %CMPROCES or %CPPROCES macro and/or by means of the %CPDUP* macros, you must implement the data checking macros as described in "Duplicate-Data Checking" on page 671 and, near the end of that topic, the case that applies to your collector or data source.

EXITSRC=*exit-source*
specifies the location where exit routines (source programs) for SAS IT Resource Management are stored. The value of this parameter can be the name of a SAS catalog that is specified in the form of *libref.catname*, or it can be a specified location in your operating environment. For UNIX or Windows, this is the complete pathname of a directory. For z/OS, this is the fully qualified name of a partitioned data set.

If you specify a SAS catalog name, then use a SAS LIBNAME statement in order to associate the *libref* with the SAS library that contains the catalog.

If you want to process your data without using exit routines, then do not specify this parameter.

For the %CSPROCES and %CWPROCES macros, observations with duplicate values for the BY list variables are removed by default. If you want to prevent the removal of duplicate observations, use the PROC180 exit point that is described in

"Exit Points for %CSPROCES and %CWPROCES" in the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

For more information on exit processing, see the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

Note: For the %CSPROCES and %CWPROCES macros, if you use process exits and you process HP OpenView Performance Agent data, then you must specify LOGFMT=SINGLETAB in order for your data to be processed. For more information, see the LOGFMT= parameter. △

GENLIB=*libref*
specifies the *libref* for the SAS library that contains the staged data that is to be processed into the PDB. The staged data has been created by user-provided software and is to be processed into a PDB by using COLLECTR=GENERIC. The data may be a SAS data set, a SAS DATA step view, a SAS SQL view, or a SAS/ACCESS view. For each table, the name of the data set, DATA step view, SQL view, or SAS/ACCESS view that has staged data for the table is specified by the table's external name parameter.

Use this parameter to specify the input data set or view for the SAS IT Resource Management process macro that you use to process your data, but only if you specify COLLECTR=GENERIC and TOOLNM=SASDS. If you specify this parameter for the %CPPROCES macro, then do not specify the RAWDATA= parameter. If you specify the GENLIB= parameter for the %CMPROCES macro or the %CSPROCES macro or the %CWPROCES macro, then do not specify the (positional) *rawdata* parameter.

JES=JES2 | JES3
specifies the job entry subsystem that is running at the site at which SMF data is collected. *The default value is JES2*; therefore, you need to specify this parameter only if your data is collected at a JES3 site. JES2 indicates that TYPE26J2 data is collected, and JES3 indicates that TYPE26J3 data is collected.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

UNIT=DISK | TAPE | CART |*other*

is the standard IBM unit specification. It specifies the type of unit on which the SMF-style log is stored. Use the UNIT= parameter only if you specify the *rawdata* parameter.

There is no default value for this parameter. If UNIT= is blank, then z/OS searches for the unit in the z/OS catalog system.

VOLSER=*volser*
specifies the volume serial number of the unit on which the *rawdata* file is stored. Use the VOLSER= parameter only if you specify the *rawdata* parameter.

WKCMPRES=YES | NO
specifies whether the SAS space compression feature is to be invoked on the WORK data sets that are used in processing data into the detail level of the PDB.

CAUTION:
NO turns off the compression logic and thus decreases CPU consumption (but increases disk space usage). △

The default is YES.

%CMPROCES Notes

Before you run %CMPROCES for the first time, review the help topic "To Use MXG." From the SAS IT Resource Management GUI for z/OS, select **Help ► To Use MXG**.

The SAS formats that are stored in PGMLIB.CPEMAIN.CMMXGLX.SOURCE are used to determine the necessary MXG macros to run for each table that is selected in the PDB. If the correct MXG macro is not found when the data is processed into the PDB, then there might be a problem with the format in PGMLIB for that MXG macro. If you experience this problem, then contact SAS Technical Support.

It is not necessary to select the component tables from which the tables XJOBS, XPRINT, XSTEPS, and XRMFINT tables are created, unless you want to keep the data from the component tables in your PDB.

A member named CMQSTART has been added to the CPMISC PDS that is built during installation of SAS IT Resource Management on z/OS. This file contains a sample batch job that allocates a PDB, runs the process and reduce tasks on a set of tables, and generates reports at the end. You can use this sample as a quick start job after you install this application.

%CMPROCES has implemented support in order to prevent future data from being inadvertently processed into a PDB and thus subsequently aging older data out of the PDB.

Note: For information about setting age limits for your data and information about the reference date for age limits, see "Specify the Age Limit for a Level in a Table" in the "Administration: Working with Levels" chapter in the *SAS IT Resource Management User's Guide*. △

By default, %CMPROCES deletes the data sets that are written to COLLECT when it has completed. The CPSTGEKP macro variable enables you to specify whether the staged data is retained in its staged location or deleted at the end of the %CMPROCES macro.

If you want to retain the staged data sets, specify the following before the %CMPROCES macro:

```
%let cpstgekp=Y;
```

If you want to delete all data in the staged location, specify the following before the %CMPROCES macro:

```
%let cpstgekp=N;
```

%CMPROCES Examples

The following example processes SMF data in the current generation of the SMF log into the XTY70, XTY71, and XTY72 tables. In this example, the logged SMF-style data is stored as part of a generation data group (GDG):

```
%cmproces( smf-gdg-name(0),
           xty70 xty71 xty72,
           collectr=smf,
           toolnm=mxg,
           _rc=retcode);
%put CMPROCES return code is &retcode;
```

The following example processes data that is not SMF into the active PDB from the SAS data library named *your.SAS.data*:

```
libname input 'your.SAS.data'
disp=shr;
%cmproces(,
     collectr=generic,
     toolnm=sasds,
     genlib=input,
     _rc=retcode) ;
 %put CMPROCES return code is &retcode;
```

%CPACCFMT

Creates formats that can be used by the %CPPROCES macro when processing UNIX accounting data

%CPACCFMT Overview

When you process UNIX accounting data with the %CPPROCES macro, %CPPROCES uses formats that map user IDs (by means of user ID numbers) to user names and group IDs (by means of group ID numbers) to group names. The %CPPROCES macro can generate the formats at run time and/or can use formats that were pre-built by the %CPACCFMT macro.

If you want to pre-build formats, for each domain run the ITSVFMT script to write representative UNIX accounting data and its corresponding password and group information to a file (formatfile). The %CPACCFMT reads these files and writes one ACCDOM format, writes one or more user formats, and writes one or more group formats. Unless these formats are written to the library referenced by the libref WORK, these formats are permanent. (That is, these formats exist until overwritten by another run of the %CPACCFMT macro.)

%CPACCFMT Syntax

%CPACCFMT(
    RAWDATA=*location-of-directory-or-PDS*

<,**_RC=***macro-var-name*>
<,**STORELOC=***libref*>);

Details

RAWDATA=*location-of-directory-or-PDS*
specifies the full path and directory name of the directory that contains files generated by the ITSVFMT script (on UNIX or Windows), or specifies the fully qualified name of the PDS that contains members generated by the ITSVFMT script (on z/OS). (If only one formatfile is to be used, the RAWDATA= parameter can specify the full path and file name of that file.) *This parameter is required.*

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

STORELOC=*libref*
specify the libref of the library to which the formats should be written. You must have write access to the library. *The default setting is STORELOC=SITELIB*.

The formats are stored in the catalog named **`libref.CPFMTS`** unless the libref is WORK, in which case the formats are stored in the catalog named **`WORK.FORMATS`**. In either case, if the catalog does not already exist, it will be created.

Note: If you use the setting STORELOC=WORK, remember that the WORK library is temporary and expires at the end of the current SAS session. Thus, the formats exist only during the SAS session in which they are created.

Note: The libref SITELIB is ideal if you are using the formats in several PDBs. If you are using the formats in only one PDB, you may prefer to activate that PDB and use the ADMIN libref.

%CPACCFMT Notes

Here is an overview of the formats:

- The ACCDOM format maps each domain to a number *nnnn*, where *nnnn* is 0001, 0002, 0003, and so on.

  For example, the ACCDOM format might map

```
domain CN to 0001
domain TW to 0002
```

- For each value of *nnnn*, there is one user format. The user format is based on the information in the passwd file that was included into that domain's format file by the ITSVFMT script. The user format is named $p*nnnn*, and maps each user ID number to its corresponding user name. (There is a one-to-one mapping between a user ID and a user ID number.)

  For example, the $P0001 format might map

  ```
  useridnumber i1 to user Austin B. Carroll
  useridnumber i2 to user Darlene E. Founder
  ```

  where *i1* and *i2* are user ID numbers representing user IDs such as **sasabc** and **sasdef**. And the **$P0002** format might map

  ```
  useridnumber i3 to user Danilo E. Foster
  useridnumber i4 to user Johnetta K. Law
  ```

  where *i3* and *i4* are user ID numbers representing user IDs such as **sasdef** and **sasjkl**.

- For each value of *nnnn*, there is one group format. The group format is based on the information in the group file that was included into that domain's format file by the ITSVFMT script. The group format is named $g*nnnn* and maps each group ID number to its corresponding group name. (There is a one-to-one mapping between a group ID and a group ID number.)

  For example, the **$G0001** format might map

  ```
  groupidnumber j1 to group Accounts Receivable
  groupidnumber j2 to group Accounts Payable
  ```

  where *j1* and *j2* are group ID numbers representing group IDs such as **are** and **apa**. And the **$G0002** format might map

  ```
  groupidnumber j3 to group New York
  groupidnumber j4 to group Chicago
  ```

  where *j3* and *j4* are group ID numbers representing group IDs such as **nyc** and **chi**.

- As a result, formats **$P0001** and **$G0001** will be available for use with data from domain **CN**, and formats **$P0002** and **$G0002** will be available for use with data from domain **TW**.

For examples of the formats, in the *SAS IT Resource Management Server Setup Guide* see the collector-specific appendix named "Enhanced UNIX Accounting Support User Guide."

Here is an overview of the steps for generating and using the pre-built formats:

1. You run the ITSVFMT script one time on each domain for which pre-built formats are to be generated. Each time you run the script, it writes a representative file that contains header, password, and group information. Use one directory to store all the files.
2. The %CPACCFMT macro reads the files in that directory and generates one ACCDOM format. Also, for each domain, the %CPACCFMT macro generates one user format (from the passwd file) and one group format (from the group file).
3. If the %CPPROCES macro finds password and group information supplied with incoming ACCTON data, %CPPROCES itself generates temporary password and group formats, and does not look for pre-built formats.

   If the %CPPROCES macro does not find password and group information supplied with incoming ACCTON data, the %CPPROCES macro uses the pre-built ACCDOM format and the pre-built password and group formats.

If the %CPPROCES macro does not find password and group information supplied with incoming ACCTON data and does not find the pre-built formats, %CPPROCES uses the password and group information on the machine on which %CPPROCES is running and generates temporary password and group formats.

For each block of ACCTON data that the %CPPROCES macro reads, %CPPROCES handles the formats independently. So temporary formats may be generated for one block, pre-built formats used for another block, and so on.

For more information about the steps, including choosing whether to generate formats at run time or to generate them ahead of time, and for more information about the ITSVFMT script, in the *SAS IT Resource Management Server Setup Guide* see the collector-specific appendix named "Enhanced UNIX Accounting Support User Guide."

Note: Running the %CPACCFMT macro will replace existing ACCDOM, user, and group formats. If you have several input files created by the 'itsvfmt' shell script, then you should place all those files in one directory and process them together. (If you, instead, process them one at a time, only the last file's password and group information would be represented in the ACCDOM, user, and group formats.)

%CPACCFMT Examples

Example 1: Running on UNIX and Passing in a Directory

This example points the RAWDATA= parameter to a directory that contains files. Each file is the output from one run of the ITSVFMT script. The %CPACCFMT macro looks at all the data in all the files. The result is one ACCDOM format and, for each domain in the data, one user format and one group format. The default value for the STORELOC= parameter is used, so the formats are stored in SITELIB.CPFMTS.

```
%CPACCFMT(RAWDATA=/data/format/);
```

Example 2: Running on z/OS

This example points the RAWDATA= parameter to a PDS in which each member is a file generated by one run of the ITSVFMT script. The formats that are generated are stored in WORK.FORMATS and exist only during the current SAS session. (Thus, the %CPPROCES macro would also need to be called in the current SAS session, so that %CPPROCES accesses the formats while they still exist.) The macro variable FMTRC, which contains the value of the macro's return code when the macro finishes, is written to the SAS log.

```
%CPACCFMT(RAWDATA='USER.FORMAT.PDS'
         ,STORELOC=WORK
         ,_RC=fmtrc);

%put CPACCFMT return code is &fmtrc;
```

%CPACCUTL

Displays and maintains operating system and release information that is used by the %CPPROCES macro when processing UNIX accounting data

%CPACCUTL Overview

The %CPACCUTL macro lists, adds, and deletes information that the %CPPROCES macro uses to process UNIX accounting data. If the %CPPROCES macro is called with the setting COLLECTR=ACCUNX, the %CPPROCES macro reads a header in the data that identifies the operating system (OS) and release of the operating system (OSR) that was running when the data was collected. Then, %CPPROCES looks up the OS/OSR pair and returns with the following information that the %CPACCUTL macro maintains:

- location of the staging code that is to be used for data that was collected on a machine running this OS/OSR pair

The %CPPROCES macro looks up the information in catalog entries named *libref*.ACCUNX.OSFMT.SOURCE, where *libref* is ADMIN, SITELIB, or PGMLIB. (The search order is ADMIN, then SITELIB, then PGMLIB. The first entry encountered is the one that %CPPROCES uses.) Each line in these catalog entries applies to one OS/OSR pair, and contains the OS ID, the OSR ID, and the three pieces of information. Lines in PGMLIB contain the default values.

SAS IT Resource Management supplies lines (and corresponding staging code) for some but not all OS/OSR pairs. The lines are in the catalog entry named PGMLIB.ACCUNX.OSFMT.SOURCE. You can use the %CPACCUTL macro to list the supplied lines. For example, suppose you run the following code from the SAS Program Editor window with SAS IT Resource Management active:

```
%CPACCUTL(function=LIST
         ,libref=PGMLIB
         ,_rc=retcode);

%put Return code is: &retcode;
```

Text such as this displays in the SAS log:

```
        ============================================
        == UNIX ACCOUNTING OS/OSREL FORMATS REPORT ==
        ============================================

REPORT FOR PGMLIB.ACCUNX.OSFMT.SOURCE
HP-UXB.10.20                              ACC001
HP-UXB.11.00                              ACC002
SunOS5.7                                  ACC003
        =============== END OF REPORT ===============

Return code is 0;
```

Detail lines contain the following information:

- The first value on a detail line is a concatenation of the OS ID and the OSR ID.

  For example, on the first detail line above, the first value is HP-UXB.10.20, which is a concatenation of operating system ID (HP-UX) and operating system release ID (B.10.20).
- The second value is the suffix of the name of the catalog entry where the staging code is located for this OS/OSR pair. For lines in *libref*.ACCUNX.OSFMT.SOURCE, the catalog entry name that is indicated by the second value is *libref*.ACCUNX.CP*suffix*.SOURCE.

  For example, for the first detail line above, the staging code is located in PGMLIB.ACCUNX.CPACC001.SOURCE.

If a line for the OS/OSR pair for which you intend to process UNIX accounting data does not appear in the catalog entry in PGMLIB, you can add your own line.

- If you want the line to apply site-wide to PDBs containing UNIX accounting data, you can use the %CPACCUTL macro to add the line to the catalog entry named SITELIB.ACCUNX.OSFMT.SOURCE.
- If you want the line to apply only to a particular PDB containing UNIX accounting data, you can use the %CPACCUTL macro to add the line to that PDB's catalog entry named ADMIN.ACCUNX.OSFMT.SOURCE. Note: The way to indicate a particular PDB is to have that PDB be the active PDB when you add the line.

The search order that %CPPROCES uses for the catalog entries containing lines for OS/OS Release pairs is:

```
ADMIN.ACCUNX.OSFMT.SOURCE (first)
SITELIB.ACCUNX.OSFMT.SOURCE
PGMLIB.ACCUNX.OSFMT.SOURCE (last)
```

Typically, a line for a particular OS/OSR pair occurs only one time in the search path. However, if a pair has more than one line in the search path, %CPPROCES uses the first one that it encounters.

%CPACCUTL Syntax

%CPACCUTL(
FUNCTION=LIST | ADD | DELETE
<,**INSRC=**n | nn>
<,**LIBREF=**PGMLIB | SITELIB | ADMIN | _ALL_>
<,**OS=***operating-system-id*>
<,**OSREL=***operating-system-release-id*>
<,**_RC=*macro-var-name***>);

Details

FUNCTION= LIST | ADD | DELETE
specifies what the macro is to do with catalog entries that contain OS/OS Release pair information. If FUNCTION=LIST, the macro is to list the contents of one or more catalog entries. If FUNCTION=ADD, the macro is to add a line to the contents of a catalog entry. If FUNCTION=DELETE, the macro is to delete a line from the contents of a catalog entry. *The FUNCTION= parameter is required.*

- If FUNCTION=LIST, no other parameters are required. The LIBREF= and RC= parameters are optional. The INSRC=, OS=, and OSREL= parameters are ignored. The default value of the LIBREF= parameter is _ALL_.
  - If LIBREF=PGMLIB, a report is generated on the contents of the catalog entry named PGMLIB.ACCUNX.OSFMT.SOURCE.
  - If LIBREF=SITELIB, a report is generated on the contents of the catalog entry named SITELIB.ACCUNX.OSFMT.SOURCE.
  - If LIBREF=ADMIN, a report is generated on the contents of the catalog entry named ADMIN.ACCUNX.OSFMT.SOURCE are listed.
  - IF LIBREF=_ALL_ or if the LIBREF= parameter is not specified, a report is generated with three sections, one on one each of the catalog entries above.

- If FUNCTION=ADD, the following additional parameters are required: INSRC=, LIBREF=, OS=, and OSREL=. The _RC= parameter is optional. There is no default value for the LIBREF= parameter. You can specify LIBREF=SITELIB or LIBREF=ADMIN.
  - If LIBREF=SITELIB, a line with the values specified by the additional parameters is added to the catalog entry named SITELIB.ACCUNX.OSFMT.SOURCE.
  - If LIBREF=ADMIN, a line with the values specified by the additional parameters is added to the catalog entry named ADMIN.ACCUNX.OSFMT.SOURCE.
- If FUNCTION=DELETE, the following additional parameters are required: INSRC=, LIBREF=, OS=, and OSREL=. The _RC= parameter is optional. There is no default value for the LIBREF= parameter. You can specify LIBREF=SITELIB or LIBREF=ADMIN.
  - If LIBREF=SITELIB, the line with the values specified by the additional parameters is deleted from the catalog entry named SITELIB.ACCUNX.OSFMT.SOURCE.
  - If LIBREF=ADMIN, the line with the values specified by the additional parameters is deleted from the catalog entry named ADMIN.ACCUNX.OSFMT.SOURCE.

INSRC=*n* | *nn*
specifies one (n) or two (nn) digits, where each digit is in the range 0–9. If only one digit (n) is specified, the macro pads it on the left with zero, so that it becomes two digits (0n).

- If FUNCTION=LIST, the INSRC= parameter is not active (that is, the INSRC= parameter is ignored).
- If FUNCTION=ADD, the value of the INSRC= parameter is used to construct a line in the catalog entry SITELIB.ACCUNX.OSFMT.SOURCE (if LIBREF=SITELIB) or a line in the catalog entry ADMIN.ACCUNX.OSFMT.SOURCE (if LIBREF=ADMIN).

  The value of the INSRC= parameter is used to construct the suffix (the third value on the line).
  - If INSRC=n, the value of suffix is ACC90n and means that the staging code for the OS/OSR pair specified by the OS= and OSREL= parameters is (or will be) in SITELIB.ACCUNX.CPACC90n.SOURCE (if LIBREF=SITELIB) or in ADMIN.ACCUNX.CPACC90n.SOURCE (if LIBREF=ADMIN).
  - If INSRC=nn, the value of suffix is ACC9nn and means that the staging code for the OS/OSR pair specified by the OS= and OSREL= parameters is (or will be) in SITELIB.ACCUNX.CPACC9nn.SOURCE (if LIBREF=SITELIB) or in ADMIN.ACCUNX.CPACC9nn.SOURCE (if LIBREF=ADMIN).

  Note: The CPACC90n or CPACC9nn catalog entry may exist or not exist at the time that you call %CPACCUTL, and that the catalog entry is *not* created by the %CPACCUTL macro. (You must create the catalog entry.) However, the catalog entry must exist and contain the staging code for OS/OSR pair specified by OS= and OSREL= by the time that you call the %CPPROCES macro to process the UNIX accounting data. △
- If FUNCTION=DELETE, the value of the INSRC= parameter is used to find the line that is to be removed from the catalog entry

SITELIB.ACCUNX.OSFMT.SOURCE (if LIBREF=SITELIB) or from the catalog entry ADMIN.ACCUNX.OSFMT.SOURCE (if LIBREF=ADMIN).

The value of the INSRC= parameter is used to construct a suffix, and the constructed suffix must match the suffix on the line to be removed.

- If INSRC=n, the constructed suffix is ACC90n.
- If INSRC=nn, the constructed suffix is ACC9nn.

Note: The names of supplied catalog entries that contain staging code take the form PGMLIB.ACCUNX.CPACC*.SOURCE, where * can range from 000 to 899. The names of user-created catalog entries that contain staging code take the form SITELIB.ACCUNX.CPACC*.SOURCE or ADMIN.ACCUNX.CPACC*.SOURCE, where * can range from 900 to 999. The range constraints are an attempt to avoid name conflict (number conflict). △

If you use both the SITELIB and ADMIN libraries for user-added catalog entries that contain staging code, it is highly recommended that you also avoid name conflicts (number conflicts) between entries in SITELIB and ADMIN, so that every entry that contains staging code has a unique number. For example, you could accomplish this by using the range 900 to 949 for SITELIB and by using the range 950 to 999 for ADMIN.

LIBREF= PGMLIB | SITELIB | ADMIN | _ALL_
specifies the libref of the library (or libraries) to which the FUNCTION= parameter applies. The value _ALL_ means PGMLIB and SITELIB and ADMIN.

If FUNCTION=ADD or FUNCTION=DELETE, you must have update access to the library that you specify.

For more information about this parameter, see the FUNCTION= parameter and the INSRC= parameter.

OS=*operating-system-id*
specifies the ID of the operating system to which this line in the catalog entry applies. Do not wrap the ID with single or double quotes.

- If FUNCTION=LIST, this parameter is not active (that is, is ignored if specified).
- If FUNCTION=ADD, this parameter is required.
- If FUNCTION=DELETE, this parameter is required.

To obtain the operating system ID, issue the UNIX command

```
uname -s
```

on the operating system from which the UNIX accounting data originates. On HP UNIX systems, the command returns the value HP-UX, so you specify OS=HP-UX.

OSREL=*operating-system-release-id*
specifies the release ID to which this line in the catalog entry applies. Do not wrap the ID with single or double quotes.

- If FUNCTION=LIST, this parameter is not active (that is, is ignored if specified).
- If FUNCTION=ADD, this parameter is required.
- If FUNCTION=DELETE, this parameter is required.

To obtain the release ID of the operating system, issue the UNIX command

```
uname -r
```

on the operating system from which the UNIX accounting data originates. For example, on an HP UNIX system, if the command returns the value B.10.20, then you specify OSREL=B.10.20.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPACCUTL Notes

For additional syntax information, see the FUNCTION= parameter.

For additional information about the numbering scheme for catalog entries that contain staging code, see the INSRC= parameter.

Typically, the staging code varies only in a minor way from one OS/OSR pair to the next. Thus, staging code in SITELIB and/or ADMIN is usually a modification of staging code in PGMLIB. To create the staging code in SITELIB or ADMIN, you can use one of the SAS editor windows to copy a catalog entry from PGMLIB, modify the code in the entry, and save the code to an entry in SITELIB or ADMIN.

For example, suppose you want to modify the staging code in PGMLIB.ACCUNX.CPACC001.SOURCE and to save the modified code in SITELIB.ACCUNX.CPACC901.SOURCE. You could invoke SAS IT Resource Management, make the SAS Program Editor window the active window, issue the command

```
COPY PGMLIB.ACCUNX.CPACC001.SOURCE
```

modify the code, and then issue the command

```
SAVE SITELIB.ACCUNX.CPACC901.SOURCE
```

%CPACCUTL Examples

Example 1: Listing the Contents of All Available OSFMT.SOURCE Entries

```
%CPACCUTL(function=LIST
         ,libref=_ALL_
         ,_rc=retcode);

%put Return code is: &retcode;
```

This call to %CPACCUTL writes to the SAS log a report that has three sections:

- one section lists the lines in catalog entry PGMLIB.ACCUNX.OSFMT.SOURCE
- one section lists the lines in catalog entry SITELIB.ACCUNX.OSFMT.SOURCE
- one section lists the lines in the active PDB's catalog entry ADMIN.ACCUNX.OSFMT.SOURCE.

The report is followed by a line that displays the return code of the call to %CPACCUTL.

Example 2: Adding an OS/OS Release Pair to the ADMIN Library

```
%CPACCUTL(FUNCTION=ADD
        , LIBREF=ADMIN
        , OS=HP-UX
        , OSREL=B.10.20
        , INSRC=1
        , _RC=&RETCODE);

%put Return code is &retcode;
```

This call to %CPACCUTL adds a line to ADMIN.ACCUNX.OSFMT.SOURCE in the active PDB. The line is for release B.10.20 of the HP-UX operating system. The staging code is in ADMIN.ACCUNX.CPACC901.SOURCE.

The result of this macro call is the addition of the following line to the SAS catalog entry named ADMIN.ACCUNX.OSFMT.SOURCE:

```
HP-UXB.10.20     ACC901
```

%CPARCRET

Retrieves archived data

%CPARCRET Overview

The %CPARCRET macro enables you to retrieve data that has been archived for one PDB and to restore the data to another empty PDB, which you must create and activate (in write mode) before you call the %CPARCRET macro. After the data is retrieved, it is loaded into the detail level of the new empty (and now active) PDB, and views for the tables are automatically built. You can use the %CPSTART macro to create and activate the empty PDB.

%CPARCRET Syntax

%CPARCRET(
ARCPDB=*pdb-name*
,**BEGIN=***datetime* | *date*
<,**END=***datetime* | *date*>
<,**_RC=***macro-var-name*>
<,**TABLES=***tablist* | _ALL_>);

Details

ARCPDB=*pdb-name*
specifies the name of the PDB that originally contained the archived data, which you want to restore. The name includes the complete directory path or high-level qualifiers and the PDB name, but it does not include the lowest libraries such as ADMIN, DAY, or any of the other libraries within the PDB. When you archive data, an archive history log is updated with information about the data that is archived, that is, what tables and date ranges each archive library contains. When you want to retrieve data, %CPARCRET reads the archive history log to determine where the archived data resides and which archive libraries to use in order to satisfy the BEGIN= and END= date criteria.

When the data from this PDB is restored, it is restored to your active PDB, which should be a new PDB that does not contain any data. One way that you can create and activate the new PDB is to specify it in the PDB= parameter in the %CPSTART macro.

This parameter is required.

BEGIN=*datetime* | *date*
specifies the earliest or beginning datetime or date of the data that you want to retrieve from the archive that is associated with the PDB that is specified by the ARCPDB= parameter. You can specify a DATETIME or DATE value. If you specify a date by using the *ddMMMyyyy* format, then a time of 00:00 is assumed. You can specify the date and time by using the *ddMMMyyyy:hh:mm* format.

The date portion of this parameter is required.

%CPARCRET retrieves data only for entire archive libraries. For example, if you specify a time with a retrieval range from noon to 3 PM, but the archive library has data from 8 AM to 5 PM, then all of the data from 8 AM to 5 PM is retrieved.

END=*datetime* | *date*
specifies the latest or last datetime or date for the range of data that you want to retrieve from the archive that is associated with the PDB that is specified by the ARCPDB= parameter.

This parameter is optional. If you specify this parameter, then you can specify a DATE or a DATETIME value. Specify the datetime using the *ddMMMyyyy:hh:mm* format. The time portion of this parameter is optional. If you specify an ending date, such as *ddMMMyyyy*, and you do not specify an ending time, then a time of 23:59:59 is assumed.

If no END datetime value is given, then %CPARCRET restores, for the specified tables, all data with datetimes after the datetime that is specified in the BEGIN= parameter. Additionally, %CPARCRET retrieves data only for entire archive libraries. For example, if you specify a time with a retrieval range from noon to 3 PM, but the archive library has data from 8 AM to 5 PM, then all the data from 8 AM to 5 PM is retrieved.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

TABLES=*tablist* | _ALL_
lists the tables whose data you want to retrieve from the archived data. If you specify a list of tables, then use one or more blanks to separate items in the list.

This parameter is not required. If you do not specify a table name, then data for all archived tables in the PDB that are specified by the ARCPDB= parameter is restored to the active PDB.

%CPARCRET Notes

For each table that is specified by the TABLES= parameter, if there is any data for that table in a given archive library in the archive, then all the data for that table in that archive library is retrieved. For example, if you request that data that is collected between noon and 3 PM be retrieved, but one library in the archive contains data collected between 8 AM and 1 PM, and another library in the archive contains data collected between 1 PM and 5 PM, then all the data that is collected from 8 AM to 5 PM is retrieved.

Because archived data is data that is read into the detail level, the retrieved data does not have summary statistics. For more information on archiving data, refer to the topic “Archive Overview” on page 667.

You can use a WHERE clause to examine the data more closely.

1. On the Administration tab of the main menu, select **Examine PDB Data**. The Examine Data window opens.
2. Highlight to select the view you want to work with.
3. Click the right mouse button to view the data. A list of the observations in that data view opens.
4. Select the cell you want to use to subset the data.
5. Click the right mouse button and select **Where** from the list of actions that are available.

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. △

%CPARCRET Example

This example retrieves all data for the XTY70 data table that was processed into the sys1.itrm.pdb and that has a datetime stamp after 01AUG2003.

```
%cparcret(tables=xty70, arcpdb=sys1.itrm.pdb,
          begin=01AUG03);
```

When the data is retrieved, it is stored in the active PDB.

%CPAVAIL

Creates a table that can be used for availability reporting

%CPAVAIL Overview

The %CPAVAIL macro creates in the active PDB a new table that is based on an existing table in the active PDB. The new table has additional variables and additional observations that make the new table suitable for reporting on availability.

The new table is different from regular tables. For more information, see the "Notes" section of this macro.

%CPAVAIL Syntax

%CPAVAIL(
INTABLE=*existing-tablename*
,**KEEP=***list-of-variables*
,**KEYVAR=***key-varname*
,**OUTABLE=***new-tablename*
<,**_RC=***macro-var-name*>
<,**TIMESTAM=***timestamp-indicator*>
<,**WHERE=***where-expression*>);

Details

INTABLE=*existing-tablename*
specifies the name of an existing table in the active PDB. The table must have a variable whose availability you want to track. (For more information about the variable, see the KEYVAR= parameter.)

The existing table must be of type INTERVAL, and the DATETIME variable must represent either the start or end of an interval, based on the value of the TIMESTAM= parameter. *This parameter is required*.

The %CPAVAIL macro reads the view that is named DETAIL.*existing-tablename*. If the value of the TIMESTAM= parameter is STARTIME, then each observation in DETAIL.*existing-tablename* represents an interval that starts at the observation's value of DATETIME and has a length of the observation's value of DURATION. Thus the interval began at the observation's DATETIME and ended at the observation's DATETIME+DURATION. The %CPAVAIL macro assumes that the existence of the observation means that the equipment was "up" (available) during that interval.

If the value of TIMESTAM= is ENDTIME, then the end time is used with DURATION in order to calculate the starting datetime of the interval.

If the next observation in DETAIL.*existing-tablename* has a value of DATETIME that is within 30 seconds of the value of this observation's DATETIME+DURATION, then the %CPAVAIL macro assumes that the equipment was "up" (available) during the gap between the observations.

If the next observation in DETAIL.*existing-tablename* has a value of DATETIME that is not within 30 seconds of the value of this observation's DATETIME+DURATION, then the %CPAVAIL macro assumes that the equipment was "down" (unavailable) during the gap between the observations.

%CPAVAIL does not work with the existing table's summary-level data. The new table's summary data is based on the new table's detail-level data, not the existing table's summary-level data.

CAUTION:

The AGELIMIT on the DETAIL level of this table must be greater than 0. If it is not greater than 0, then any outages that are not completely contained within a single CPREDUCE run will not be properly recorded. If you run CPREDUCE and the last record was the beginning of an outage, and then the next run of CPREDUCE has the first record after the system came back up, then that record will show up as an outage if AGELIMIT=0. △

KEEP=*list-of-variables*

lists one or more variables that are to be in the new table (in addition to DATETIME, DURATION, DATE, TIME, UPTIME, AVAIL, UNAVAIL, and all variables in the existing table's BY variables list, CLASS variables lists, and ID variables lists). If there is more than one variable, then separate the variables with one or more spaces. The variables must have a Kept status of YES in the existing table's view DETAIL.*existing-tablename*. The variables must be identifiers (not measurements).

This is an optional parameter. If you do not specify this parameter, then no additional variables are added.

KEYVAR=*key-varname*

specifies the name of a variable in the table that is specified by the INTABLE= parameter (the name of the variable whose availability you want to track). *This variable must be available in the BY variables list and the CLASS variables lists in the existing table (specified in the INTABLE= parameter).* If it is not in all these lists, then the %CPAVAIL macro terminates. Typically, this variable is the one that is immediately to the left of the DATETIME HOUR SHIFT variables in the existing table's BY variables list and CLASS variables lists.

OUTABLE=*new-tablename*

specifies the name of the new table that will be created (in the active PDB), based on the table specified by the INTABLE= parameter. The table name should begin with the letter U to indicate a user-defined table. The new table will be of type INTERVAL. This table should not already exist. If the table does exist, then this call to the %CPAVAIL macro will be ignored. This parameter is required.

The BY variables list for the new table is copied from the BY variables list of the existing table. The CLASS variables lists for the new table are copied from the BY variables list of the new table, except that the HOUR variable is removed from the CLASS variables lists if HOUR is in the BY variables list.

Detail-Level Data:

The %CPAVAIL macro creates the view that is named ADMIN.*new-tablename*. The detail-level data in ADMIN.*new-tablename* is based on the existing table's view named DETAIL.*existing-tablename*. The observations in the view ADMIN.*new-tablename* are based both on the observations in the view DETAIL.*existing-tablename* and on the gaps between the observations in the view DETAIL.*existing-tablename*. Here are the three cases:

1 For each observation in the view DETAIL.*existing-tablename*, the view ADMIN.*new-tablename* provides a corresponding observation. The new table's observation has the following variables with values as described:

DATETIME

specifies a variable whose value is a copy of the existing observation's value of DATETIME.

DURATION

specifies a variable whose value is a copy of the existing observation's value of DURATION.

DATE
is a formula variable. The value is a copy of the date part of DATETIME.

TIME
is a formula variable. The value is a copy of the time part of DATETIME.

UPTIME
is the amount of time that the equipment was "up" (available). Because the existence of the observation is assumed to mean that the equipment was "up" during the interval, UPTIME=DURATION.

AVAIL
is a formula variable. The value is the percent of the interval that the equipment was "up" (available). Because the existence of the observation is assumed to mean that the equipment was "up" during the interval, AVAIL=100%.

UNAVAIL
is a formula variable. The value is the percent of the interval that the equipment was not "up" (unavailable). Because the existence of the observation is assumed to mean that the equipment was "up" during the interval, UNAVAIL=0%.

- Every other variable that has a Kept status of YES in the existing table and satisfies at least one of the following criteria:
  - it is in the existing table's BY variable list
  - it is in one or more of the existing table's CLASS variables lists
  - it is in one or more of the existing table's ID variables lists
  - it is in the list specified by the KEEP= parameter in this call to the %CPAVAIL macro.

  The values of the variables are copies of the existing observation's values of the variables.

2 If the gap between observations in the view DETAIL.*existing-tablename* is less than 30 seconds, then %CPAVAIL assumes that the equipment was "up" (available) during the gap, and as a result the gap is ignored. No observation in the view ADMIN.*new-tablename* corresponds to the gap in the view DETAIL.*existing-tablename*.

3 If the gap between observations in the view DETAIL.*existing-tablename* is 30 seconds or more, then %CPAVAIL makes an additional observation to represent the gap. The additional observation has the following variables with values as described:

DATETIME
is a variable whose value identifies the start of the gap. DATETIME (of the gap observation) equals DATETIME (of the before-the-gap observation) plus DURATION (of the before-the-gap observation).

DURATION
is a variable whose value identifies the length of the gap. DURATION (of the gap observation) equals DATETIME (of the after-the-gap observation) minus DATETIME (of the gap observation).

DATE
is a formula variable. The value is a copy of the date part of DATETIME (of the gap observation).

TIME
is a formula variable. The value is a copy of the time part of DATETIME (of the gap observation).

UPTIME
is the amount of time that the equipment was "up" (available). Because the gap is longer than 30 seconds, the %CPAVAIL macro assumes that the equipment was not "up" during the gap, so UPTIME=0.

AVAIL
is the percentage of the interval that the equipment was "up" (available). Because the gap is longer than 30 seconds, the %CPAVAIL macro assumes that the equipment was not "up" during the gap, so AVAIL=0%.

UNAVAIL
is the percentage of the interval that the equipment was not "up" (unavailable). Because the gap is longer than 30 seconds, the %CPAVAIL macro assumes that the equipment was not "up" during the gap, so UNAVAIL=100%.

- Every other variable that has a Kept status of YES in the existing table and satisfies at least one of the following criteria:
  - it is in the existing table's BY variables list
  - it is in one or more of the existing table's CLASS variables lists
  - it is in one or more of the existing table's ID variables lists
  - it is in the list specified by the KEEP= parameter in this call to the %CPAVAIL macro.

  The values of the variables are copied from the values in the before-the-gap observation.

The BY variables list for the new table is the same as the BY variables list for the existing table.

Summary-Level Data:

For the new table, the reduce step summarizes data in ADMIN.*new-tablename* and writes or updates the summary statistics in the summary levels. Thus, the data in the summary levels is based not only on the observations from DETAIL.*existing-tablename*, but also on the observations that %CPAVAIL added to represent the long (greater than 30 seconds) gaps between observations.

The CLASS variables lists in the summary levels of the new table are the same as the CLASS variables lists in the summary levels of the existing table (except that the HOUR variable, if present in the existing table's lists, is removed from the new table's lists). The ID variables lists in the summary levels of the new table are the same as the ID variables lists in the summary levels of the existing table. The BY variables list for the new table is the same as the BY variables list for the existing table.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

TIMESTAM=*timestamp-indicator*
specifies whether the value of DATETIME (in the incoming data) represents the beginning or the end of the interval that is represented by the observation. Valid values are STARTIME *(the default)* and ENDTIME.

STARTIME indicates that the incoming DATETIME values represent the start of the interval.

ENDTIME indicates that the DATETIME value represents the end of an interval. If you specify TIMESTAM=ENDTIME, then DURATION is subtracted from the DATETIME value in order to determine the DATETIME value in the new table. The DATETIME value in the new table (OUTABLE) is always a STARTIME value.

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in ('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

%CPAVAIL Notes

If the %CPAVAIL macro detects that the new table already exists in the active PDB, then the macro terminates without affecting the new table. In this case, the return code is 0.

The new table does not have detail-level data of its own. In the active PDB's ADMIN library, the new table has a view that combines some of the variables in the detail level of the existing table with some additional variables that are for use in reporting on availability. Thus, the process step appears to update the detail-level data in the new table when it processes incoming data into the detail level of the existing table. (If you specify the new table's name when you process incoming data into the active PDB, then you will get an error.)

The new table has day-, week-, month-, and year-level statistics of its own. In order to generate the statistics, the reduce step needs to obtain the detail-level data from the view in the ADMIN library instead of a view in the DETAIL library. A regular call to the %CPREDUCE macro (that is, a call that does not specify the DETAIL= parameter) uses views in the DETAIL library to access detail-level data. A regular call to the %CPREDUCE macro ignores the new table, even if the table name is in the table list parameter, because the regular call looks for the detail-level data in the DETAIL library, and there are no observations there. Thus, for the new table you must have a call to the %CPREDUCE macro that specifies DETAIL=ADMIN and lists the new table in the table list.

You can report on detail-level data and summary-level data in the new table. To report on detail-level data, specify a level of *admin* instead of a level of *detail*. In the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and the Windows environments, you can specify a level of *admin* by following this path:

Select Table ▶ Level OTHERS ▶ SAS Data Library ▶ Admin

In batch mode, specifying a level of *admin* requires a libref of ADMIN. (The libref already exists if you called the %CPSTART macro with the appropriate PDB as active PDB.) The report definition's BY variables list must include at least the variable that is specified by the KEYVAR= parameter and must include all variables that are to the left of that variable in the new table's BY variables list. The order of these variables must be unchanged. To report on summary-level data, specify a level of day, week, month, or year, as usual.

CAUTION:

Use the WHERE clause carefully so that you do not skip any data due to non-continuous values. △

The WHERE clause should subset data based on an entire subsystem or system. Consider the following. Suppose you have a table with data for two systems: CICSPROD and CICSTEST. Each of them will have continuous data if they are up. To avoid tracking the availability of the CICSTEST system, you can use a WHERE clause such as: WHERE SYSTEM="CICSPROD". On the other hand, do not use a WHERE clause based on a numerical value such as PCTBUSY<=90%. This causes observations to be dropped if they are obtained when PCTBUSY <= 90%. In that case, %CPAVAIL indicates that the system was not up for those periods, when in reality, it was available.

As a general rule, your WHERE clause should only reference variables that are in your BY list for that table.

%CPAVAIL Example

In true batch mode, you can use calls to the %CPFAILIF macro. Elsewhere, replace them with calls to the %CPDEACT macro.

```
/*
Activate a PDB for use with HP OVPA data.
*/

%cpstart ( ...
              , pdb=pdb-ovpa
              , access=write
              , _rc=myrc
              , ... );
%cpfailif (myrc ne 0);

/*
Create an availability table named USYSAVL based on the HP OVPA
global data table named PCSGLB.  Designate the variable MACHINE as
the indicator of availability.  The call to %CPAVAIL is needed only
one time.  You can remove it after the program first runs.
(If you leave the call to %CPAVAIL in the program during subsequent runs,
it will be ignored.)
*/
%cpavail ( intable = pcsglb
           , outable=usysavl
           , keyvar=machine
           , _rc=myrc );
%cpfailif (myrc ne 0);

/*
Process incoming data into the detail level of the PCSGLB table.
*/  %csproces ( /my/ovpadata
              , pscglb
              , collectr=hp-ovpa
              , _rc=myrc);
%cpfailif (myrc ne 0);
/*
Reduce data from the detail level to the day, week, month,
and year levels, in both the PCSGLB and USYSAVL tables.
*/
%cpreduce ( pcsglb
           , _rc=myrc );
%cpfailif (myrc ne 0);
%cpreduce ( usysavl
          , detail=admin
          , _rc=myrc );
%cpfailif (myrc ne 0);
/*
Generate a report based on the "detail" level of the new
table.
```

```
*/
%cpcchrt( usysavl        * name of table ;
        , avail,         * name of variable ;
        , stackvar=hour
        , by=machine date
        , redlvl=admin      * name of library ;
        , tabtype=interval
        , palette=pgmlib.palette.greenred
        , outdesc=hourly system availability
        , axis=0 to 1 by .1
        , type=vbar
        , yval=26
        , labels=(machine="Machine"
                  date="Date"
                  hour="Hour"
                  avail="System Up")
         );

/*
Generate a report based on a reduction level (month)
of the new table.
*/
%cpcchrt( usysavl            * name of table ;
        , avail,             * name of variable ;
        , stackvar=date
        , by=machine shift
        , redlvl=month       * name of library ;
        , tabtype=interval
        , palette=pgmlib.palette.greenred
        , outdesc=monthly system availability
        , axis=0 to 1 by .1
        , type=vbar
        , yval=26
        , formats=(date=monyy.)
        , labels=(machine="Machine"
                  shift="Shift"
                  date="Month"
                  avail="System Up")
        );
```

%CPBATCH

Executes the tasks on the list of tasks that are selected by using the Batch Utility (z/OS)

%CPBATCH Overview

%CPBATCH runs the tasks on the list that you prepared by using the Batch Utility within the SAS IT Resource Management z/OS GUI. If you want to use the %CPBATCH macro, then you must submit a batch job that contains at least the %CPSTART macro and this macro. This job should run after the SMF dump (or the

equivalent if you are using style records other than SMF), which typically runs at night as part of a z/OS scheduling system.

%CPBATCH Syntax

%CPBATCH(
ENV=BACK
<,**_RC=***macro-var-name*>);

Details

ENV=BACK
specifies whether the macro should run in the background. *The only value (and the default value) for this parameter is BACK*, which indicates that the macro should run in the background.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPBATCH Notes

You might want to use %CPBATCH to run the process task, the reduce task, and any reporting tasks. Or you could process and reduce your data and then run your reports at a later time by using %CPBATCH. The SAS log and output from any of these tasks can be distributed to individual users if you specify this option through the Batch Utility by selecting the MAILMAN (Mail Distribution Manager) task.

To use the %CPBATCH macro, update access to SITELIB is required. Therefore, the SITELIB= parameter must be specified on the %CPSTART macro and the SITEACC= parameter must be set to OLD.

The messages that would have gone to the LOG window are redirected to SITELIB.CPUPGMS.task_name.LOG. The text that would have gone to the OUTPUT window is redirected to SITELIB.CPUPGMS.task_name.OUTPUT. The graphs that would have gone to the GRAPH window are redirected to the catalog that is specified by the task's (report macro's) OUTLOC= parameter.

%CPBATCH Example

This example runs tasks that have been selected by using the Batch Utility within the SAS IT Resource Management server software on z/OS.

```
%cpstart(mode=batch,
         root= root-location,
         pdb= pdb-name,
         disp=old,
         sitelib= site-library,
         siteacc=old,
         mxglib= mxg.mxg.formats,
         mxgsrc=('mxg.userid.sourclib' 'mxg.mxg.sourclib'),
         _rc=retcode
         );

 %cpbatch(env=BACK);
```

%CPC2RATE

Converts a variable type from counter to rate

%CPC2RATE Overview

%CPC2RATE is used only for data that will be processed by the Generic Collector Facility, in SAS/CPE or in SAS IT Resource Management 1.

%CPC2RATE creates an output SAS view that converts the values of the variables of data type C2RATE or D2RATE from counters or deltas into rates. CPC2RATE will process either counters or deltas in a single invocation (but not both). If DURATION is missing, then the macro also sets the value of DURATION to be the difference between successive values of the variable DATETIME within one BY group.

Note: With SAS IT Resource Management 2, the data manipulation that is done by %CPC2RATE is no longer necessary in order for data to be read by %CPPROCES, because the conversion of counters and deltas to rates is now automatically performed as part of the Generic Collector Facility. △

%CPC2RATE Syntax

%CPC2RATE(
IN=*libref.name*
,**OUTVIEW=***libref.name*
,**TABNAME=***tablename*
<,**CONVERT=**C2RATE | D2RATE>
<,**_RC=***macro-var-name*>
<,**SORTED=**NO | YES>);

Details

IN=*libref.name*

specifies the libref of the library and the name of the input SAS data set or view that contains data in the form of counters or deltas. The libref must already be defined by the time that this macro is called.

OUTVIEW=*libref.name*
specifies the libref of the library and the name of the output SAS view that is to have the corresponding data in the form of rates. The libref must already be defined by the time that this macro is called.

The name of this view must not be the same as the name of the data set or view specified for the IN= parameter. As input to %CPPROCES, use the view name that you specify here; do not use the name in the IN= parameter.

The name can be as long as seven characters. The first character must be a letter or an underscore. The remaining characters can be letters, numbers, or underscores.

TABNAME=*tablename*
specifies the name of the table for which to look up the list of BY variables. The BY variables list is used to sort the input data and to determine BY groups for setting DURATION and for converting counters to rates.

This table is also used to look up the list of variables that have interpretation of C2RATE or D2RATE. This second list of variables is used to determine which variables are counters or deltas.

CONVERT=C2RATE | D2RATE
specifies which conversion is to occur. The value C2RATE indicates that counter-to-rate conversion is to occur on variables with interpretation C2RATE. The value D2RATE indicates that delta-to-rate conversion is to occur on variables with interpretation D2RATE. *The default is C2RATE*.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

SORTED=NO | YES
specifies whether the incoming data is already sorted appropriately.

- *NO* indicates that a sort is required (by using the BY variables list from the archive PDB's data dictionary for the table that is specified by the TABNAME= parameter).
- *YES* indicates that the sort is to be bypassed (that is, the incoming data is already in the proper order).

The default is NO.

%CPC2RATE Example

This example stages the data, converts counters to rates, processes the data by using the Generic Collector Facility, and reduces the data.

```
%CPSTART( pdb=/usr/tmp/my-pdb,
    access=write,
    mode=batch );

* Stage rawdata into SAS data set per generic collector rules;
data MYLIB.TAB / view=MYLIB.TAB;
  input ... ;
  run;

* Create a view that converts counters to rates;
%CPC2RATE( tabname=UTABLE,
    in=MYLIB.TAB,
    _rc=retcode,
    outview=MYLIB.UTABLE );
%put Return code is &retcode;

* Input to CPPROCES is the output view from CPC2RATE;
%CPPROCES( UTABLE,
    collectr=GENERIC,
    toolnm=SASDS,
    genlib=MYLIB );
%CPREDUCE ;
```

%CPCAT

Copies control statements to and from SAS catalog source entries

%CPCAT Overview

%CPCAT is used to copy control statements to or from a SOURCE entry in a SAS catalog. You can copy control statements from a SAS catalog source entry only if you copy them to a SAS data set or to the SAS log. This macro is useful when it is used in conjunction with the %CPDDUTL macro because, after copying instream statements into a SOURCE catalog, the SOURCE can be used as input to %CPDDUTL.

For more information about %CPDDUTL control statements, see “Using the %CPDDUTL Macro” on page 567.

%CPCAT Syntax

%CPCAT(
    <**CAT=***SAS-source-catalog-entry*>
    <,**ODS=***output-SAS-data-set* | LOG>);

Details

CAT=*SAS-source-catalog-entry*
is the four-level name (libref.catalog-name.entry-name.source) of the input or output SAS source catalog entry, depending on whether ODS= is specified.

CAT= is used to specify the output SAS source catalog entry when you want to copy instream code to a SOURCE catalog entry, as shown in the following example:

```
%CPCAT ; CARDS4 ;
... any lines of code, except no ;;;; in columns 1 through 4...
;;;;
%CPCAT (CAT=output-SAS-source-catalog-entry);
```

In the previous example, the CAT= parameter is used to specify the output source catalog, and in this case you cannot use the ODS= parameter.

The CARDS4 statement indicates that data lines that may contain semicolons follow the statement. After the last data line, a line consisting of four semicolons in bytes 1 through 4 signals the end of the data. The macro reads the lines of data, including the internal semicolons, until it reaches the line of four semicolons.

CAT= is used to specify the input SAS source catalog entry when you copy *from* a SOURCE catalog entry *to* a SAS data set or the SAS log, such as

```
%CPCAT (CAT=input-SAS-source-catalog-entry,
  ODS=output-SAS-data-set);
```

In this example, the value of CAT= is used to specify the input SOURCE catalog, and ODS= is used to specify the output location. (For an example with ODS=LOG, see the examples for this macro.)

If ODS= is not specified, then the code that is specified within the prior CARDS4 statement is written to the SOURCE catalog entry that is specified in the CAT= parameter.

ODS=*output-SAS-data-set* | LOG
specifies the destination of the statements that are being copied. It can be either a SAS log or the name of a SAS data set. Specify this parameter if you are copying a SOURCE catalog entry to either a SAS data set or the SAS log. String variables in the output data set have a maximum length of 200 characters.

%CPCAT Notes

When you use %CPCAT to copy instream data to a SAS source catalog entry, you must invoke %CPCAT twice. The first invocation specifies the instream data, and the second invocation specifies the name of the output SAS source catalog entry, as shown in the following example:

```
%CPCAT ; CARDS4 ;
  any lines of code, except no ;;;; in columns 1 through 4
;;;;
%CPCAT (CAT=output-SAS-source-catalog-entry);
```

%CPCAT Examples

The following example copies ADD TABLE control statements to a SAS catalog member SAS-CATALOG-SOURCE and then invokes %CPDDUTL to execute the statements. When you use the CARDS4 statement, the four semicolons (;;;;) must appear alone and at the beginning of a line.

```
%cpcat; cards4 ;
   add table
       name=tab01 ;
   add table
       name=tab02 ;
;;;;
%cpcat(cat=sas-catalog-source-entry) ;
%cpddutl(entrynam=sas-catalog-source-entry,list=Y) ;
```

The following example copies a source catalog entry to an output data set and prints the data set:

```
%cpcat(cat=sasuser-test-source-entry,
       ods=work.temp) ;
proc print data=work.temp noobs uniform ;
  var card ;
run ;
```

The following example copies a source catalog entry to the SAS log:

```
%cpcat(cat=sasuser-test-source-entry, ods=LOG) ;
```

%CPDBCOPY

Copies the contents of one PDB to another PDB

%CPDBCOPY Overview

The %CPDBCOPY macro copies the contents of all nine libraries in one PDB to the corresponding libraries in another PDB, with these exceptions:

- The value of the archive path option (in the DICTLIB library) is not copied.
- The PDB's archive libraries (if any) and **qs** directory or .QS PDS (if any) are not copied.

Note: %CPDBCOPY erases the data dictionary and data in the destination PDB before it begins copying. △

%CPDBCOPY Syntax

%CPDBCOPY(
pdbfrom
,*pdbto*
<,**_RC=***macro-var-name*>);

Details

pdbfrom
is the name of the source PDB. The name includes the complete directory path or high-level qualifiers and the PDB name. It does not include the lower libraries such as ADMIN, DAY, or any others.

pdbto
is the name of the target PDB. This value includes the complete directory path or high-level qualifiers and the PDB name, but it does not include the lowest libraries such as ADMIN, DAY, or any others.

If your SAS IT Resource Management server is running on UNIX or Windows, then the directory structure for this PDB must exist before you run %CPDBCOPY. You can create the directory structure through the SAS IT Resource Management GUI or by using the **mkdir** or **md** command.

If your SAS IT Resource Management server is running on z/OS, then the *pdbto* PDB must be created before you run %CPDBCOPY. You can create the PDB through the SAS IT Resource Management GUI or by using the CMBPDALC member in the *install-prefix*.CPMISC PDS, which is set up at installation time. For the complete physical location of this PDS at your site, see your site administrator or your SAS Installation Representative.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPDBCOPY Notes

The *pdbfrom* can be the active PDB, which is the active PDB specified in the PDB= parameter in the %CPSTART macro. The *pdbto* cannot be the active PDB.

CAUTION:
In Windows, %CPDBCOPY will fail if any directory in the *pdbto* is currently in use. A directory is considered to be "in use" if it is the selected directory in Windows Explorer or if it is the active directory in an MS-DOS command window. Change directories to another location or exit any Windows Explorer or MS-DOS window before using %CPDBCOPY. △

%CPDBCOPY Example

This z/OS example copies all the data dictionary information and data in YOUR.PDB1 to YOUR.PDB2.

```
%cpdbcopy(your.pdb1,
          your.pdb2);
```

%CPDBKILL

Deletes the data and tables from a PDB, and deletes the contents of the PDB's data dictionary

%CPDBKILL Overview

The %CPDBKILL macro deletes everything within each of the PDB's nine libraries and then establishes an empty data dictionary that is ready for use. However, the PDB's archive libraries, if there are any, and other miscellaneous objects (such as the *qs* directory or *.QS* PDS, if the QuickStart Wizard was used) are not emptied by this macro.

%CPDBKILL Syntax

%CPDBKILL(
pdb-name
<,**_RC=***macro-var-name*>);

Details

pdb-name
is the name of the PDB from which the tables and data are to be deleted. The name includes the complete directory path or high-level qualifiers and the PDB name, but it does not include the lower libraries such as DETAIL, DAY, or any of the other libraries within the PDB.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPDBKILL Notes

If you want to delete only the data from all tables in a PDB, then you can use the %CPDBPURG macro (see "%CPDBPURG" on page 80). If you want to purge data only

from a specific table in the PDB, then use the %CPDDUTL control statement PURGE TABLE (see "PURGE TABLE" on page 619). If you want to delete both the data in a specific table and the table's definition, then use the %CPDDUTL control statement DELETE TABLE (see "DELETE TABLE" on page 601).

%CPDBKILL Example

This example deletes all the tables and data in the PDB referred to as *cpe-pdb*, but it keeps the structure of the data dictionary in the PDB:

```
%cpdbkill(cpe-pdb);
```

%CPDBPURG

Purges data from a PDB

%CPDBPURG Overview

The %CPDBPURG macro deletes the data from all levels of all tables in a PDB, but leaves the PDB otherwise intact.

You can use the KEEP= parameter to keep the detail-level data sets when you delete the data from the other levels. Similarly, you can use the DELETE= parameter to delete detail-level data and keep the data from the other levels.

%CPDBPURG Syntax

%CPDBPURG(
pdb-name
<,**DELETE=**DETAIL | ALL>
<,**KEEP=**DETAIL | NONE>
<,**_RC=***macro-var-name*>);

Details

pdb-name
is the name of the PDB that contains the data to be deleted. This includes the complete directory path or high-level qualifiers, but it does not include the lower libraries, such as DETAIL, DAY, or any of the other libraries within the PDB.

You can purge the active PDB. However, on z/OS you must first set the DISP= parameter to OLD when you submit the %CPSTART macro.

DELETE=DETAIL | ALL
specifies whether %CPDBPURG is to delete only the data in the detail level. If you specify DELETE=DETAIL, then %CPDBPURG deletes all data at the detail level, but it keeps the performance data at the other levels in the PDB. If you specify DELETE=ALL, then %CPDBPURG deletes the data from all levels of all tables in the active PDB. *The default is DELETE=ALL*. You can specify either this parameter or the KEEP= parameter, but not both.

KEEP=DETAIL | NONE

specifies whether %CPDBPURG is to delete all data in the DETAIL library. If you specify KEEP=DETAIL, then %CPDBPURG keeps all the detail-level data and purges the rest of the performance data. If you specify KEEP=NONE, then %CPDBPURG deletes the data from all levels of all tables in the active PDB. *The default is KEEP=NONE*. You can specify either this parameter or the DELETE= parameter, but not both.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPDBPURG Notes

Some objects that are related to the PDB are not purged by this macro. For example, the following objects are not purged: the archive libraries, if any, for the PDB; the QS directory or .QS PDS, if the PDB was created by using the QuickStart Wizard; and the ADMIN library (which may include report, rule, and palette definition folders, Web browser and platform identification tables, duplicate-checking control data sets, a portion of the availability tables, if any, and so on).

If you want to delete the tables, data, and contents of the data dictionary but you want to keep the structure of the PDB, such as the libraries and directories, then use the %CPDBKILL macro (see "%CPDBKILL" on page 79). If you want to purge data from only a specific table in the PDB, then use the %CPDDUTL control statement PURGE TABLE (see "PURGE TABLE" on page 619). If you want to delete both the data in a specific table and the table's definition, then use the %CPDDUTL control statement DELETE TABLE (see "DELETE TABLE" on page 601).

You can purge the active PDB. However, to do so on z/OS, you must set the DISP= parameter to OLD when you submit the %CPSTART macro.

%CPDBPURG Example

This example deletes the data at all levels in all tables in the PDB named *cpe-pdb*:

```
%cpdbpurg(cpe-pdb);
```

%CPDD2RPT

Creates an HP OpenView Performance Agent report parameter file from table and variable definitions in a PDB (UNIX and Windows only)

%CPDD2RPT Overview

The %CPDD2RPT macro generates an HP OpenView Performance Agent (HPOVPA) report parameter file from SAS IT Resource Management table and variable definitions.

Typically, the table and variable definitions are for data in current HPOVPA format, and you generate a report parameter file for current HPOVPA data. (However, you can use table and variable definitions for data in the old HP Performance Collection Software (HPPCS) format, and you can generate a report parameter file for old HPPCS data.)

This report parameter file (reptfile) can then be used with the HP OpenView Performance Agent's extract command to extract only the metrics required to populate the SAS IT Resource Management tables with HPOVPA (or HPPCS) data.

%CPDD2RPT Syntax

%CPDD2RPT(
PDB=*PDB-location*
,**REPTPCS=***location-of-HPPCS-report-parameter-file*
,**REPTMWA=***location-of-HPOVPA-report-parameter-file*
<,**_RC=***macro-var-name*>);

Details

PDB=*PDB-location*
specifies the name of the PDB that contains SAS IT Resource Management table definitions (and their associated variable definitions) that you want to populate with HPOVPA (or HPPCS) data. This name includes the complete directory path or high-level qualifier and the PDB name, but it does not include the lowest levels such as ADMIN, DAY, or any of the other libraries within the PDB.

This parameter is required.

Note that this PDB may be the active PDB (as specified in a call to %CPSTART) or may be a non-active PDB.

REPTPCS=*location-of-HP-PCS-report-parameter-file*
specifies the complete path and name of the file to which a report parameter file for HPPCS data is to be written. If you specify this parameter, be sure that the tables and variables in the PDB are for HPPCS data.

If you specify this parameter, do not specify the REPTMWA= parameter.

Because data in HPPCS format is old, this parameter is rarely used.

REPTMWA=*location-of-HP-OVPA-report-parameter-file*
specifies the complete path and name of the file to which a report parameter file for HPOVPA data is to be written. When you specify this parameter, be sure that the tables and variables in the PDB are for HPOVPA data.

If you specify this parameter, do not specify the REPTPCS= parameter.

Because data in HPOVPA format is current, this parameter is usually used.

_RC=*macro-var-name*

specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPDD2RPT Notes

Because you are likely to be working with current data, you are likely to use the REPTMWA= parameter instead of the REPTPCS= parameter.

You may want to generate the report parameter file from supplied table and variable definitions that you added to the PDB from the master data dictionary and have not changed. Or you may want to add the supplied table and variable definitions to the PDB, modify them, and then generate the report parameter file from the modified table and variable definitions.

Typically, you would

1. call the %CPSTART macro to establish the SAS IT Resource Management environment and allocate a new (empty) PDB and activate the PDB.
2. call %CPDDUTL to add the appropriate table definitions (and their variable definitions) from the master data dictionary to the PDB.
3. optionally, create a file of %CPDDUTL control statements to modify the tables and variables, and then call %CPDDUTL to apply the control statements to the PDB. For example, you may want to change the Kept status of some of the variables.
4. call %CPDD2RPT to generate the corresponding report parameter file.

The result is a report parameter file and PDB that are ready for production use.

%CPDD2RPT Example

```
*...;
* Edit the values below;
*...;

* Set PDBLOC to the pathname of your PDB;
%let PDBLOC=/your/pdb;

* Set REPTPCS to name of the HPPCS report parm file to create;
%let REPTPCS=/your/reptfile.pcs;
```

```
* Set REPTMWA to name of the HPOVPA report parm file to create;
%let REPTMWA=/your/reptfile.mwa;

*...;
* End of user editable values;
*...;

*...;
* Advertise;
*...;
%put NOTE: PDBLOC is &PDBLOC;
%put NOTE: REPTPCS is &REPTPCS;
%put NOTE: REPTMWA is %REPTMWA;

*...;
* Create the report parm files;
*...;
%CPDD2RPT( reptpcs=&REPTPCS,
           reptmwa=&REPTMWA,
           pdb=%PDBLOC,
           _rc=retc );

%CPFAILIF (&RETC ne 0 );
```

%CPDEACT

Deactivates a PDB

%CPDEACT Overview

The %CPDEACT macro evaluates an expression, and if the expression is evaluated as true, then %CPDEACT deactivates the active PDB. Neither the SAS IT Resource Management session nor the SAS session in which SAS IT Resource Management is running is terminated. Once the PDB is deactivated, subsequent SAS IT Resource Management macros and statements will terminate, until you specify a %CPSTART macro to activate a different PDB (or the same PDB).

%CPDEACT Syntax

%CPDEACT(*expression*);

Details

expression
specifies the expression that you want to evaluate. This expression must be a valid SAS expression and must use symbols such as GT, GE, LT, LE, EQ, or NE instead of special symbols (>, >=,<,<=, =, and ^=). Code the expression so that it evaluates as "true" if there is a problem and as "false" if there are no problems.

If the expression evaluates as true, then %CPDEACT deactivates the active PDB, but it does not terminate the SAS IT Resource Management session or the

SAS session, and control passes to the next statement. From this point forward, all SAS IT Resource Management macro calls that require an active PDB will fail until the program encounters a call to %CPSTART that activates a different (or reactivates the same) PDB.

If the expression evaluates as false, then %CPDEACT passes control to the next statement.

This parameter is required.

%CPDEACT Notes

CAUTION:

If you submit this macro through the SAS Program Editor window when the SAS IT Resource Management GUI is active, then %CPDEACT cannot deactivate the active PDB. You can submit this macro in true batch mode, or you can submit it through the SAS Program Editor window (by selecting **Locals ▶ Submit**) when the SAS IT Resource Management GUI is NOT active. If you submit this macro and %CPDEACT cannot deactivate the active PDB (as stated above), then the code that follows this macro will execute (or attempt to execute) regardless of the value of the %CPDEACT expression. (In this case, SAS is not terminated when the condition is true.)

If you wish to prevent this situation, close your SAS IT Resource Management GUI before you submit (by using the SAS Program Editor window) any code that contains a call to the %CPDEACT macro. △

In true batch mode, you can use the %CPFAILIF macro if you want to terminate the SAS IT Resource Management session and the SAS session, rather than suppress some "calls." For more information see "%CPFAILIF" on page 120.

%CPDEACT Example

The first call to %CPSTART starts a SAS IT Resource Management session, activates a PDB, and does other activities. No call to %CPDEACT is needed after the call to %CPSTART, because %CPSTART deactivates the PDB if it does not complete successfully. (%CPSTART is the only macro other than %CPDEACT that deactivates a PDB because of unsuccessful completion.)

Control passes to the call to %CPPROCES regardless of the status of completion of %CPSTART. However, %CPPROCES requires an active PDB. If %CPSTART had a problem, then %CPPROCES will not run. If %CPSTART completed successfully, then %CPPROCES will run.

The following call to %CPDEACT checks whether the call to %CPPROCES completed successfully. (A return code of zero represents successful completion; a nonzero return code represents a problem.) If the value of the return code is not zero (that is, if %CPPROCES had a problem), then %CPDEACT deactivates the PDB. If the value of the return code is zero (that is, if %CPPROCES completed successfully), then %CPDEACT does not deactivate the PDB.

The remaining pairs operate the same way: run if the PDB is still active, and deactivate the PDB if the run is not successful.

%CPRUNRPT does not have the _RC parameter, so it will pass control to the next statement and will leave the status of the PDB unchanged, regardless of the status of its completion.

When the second %CPSTART is encountered, the status of the first %CPSTART's PDB is ignored. The second %CPSTART runs. The remaining code operates in the same way as the code starting at the first %CPSTART. The remaining code is affected by the status of the PDB in the second call to %CPSTART.

```
 ...
%CPSTART ( ... , _rc=myrc) ;
* %CPPROCES will not run if PDB was deactivated ;
%CPPROCES (..., _rc=myrc )  ;
* deactivates PDB if %CPPROCES not successful ;
%CPDEACT ( &myrc ne 0 ) ;
* %CPREDUCE will not run if PDB was deactivated ;
%CPREDUCE (..., _rc=myrc) ;
* deactivates PDB if %CPREDUCE not successful ;
%CPDEACT ( &myrc ne 0 ) ;
%CPRUNRPT (...) ;
...
%CPSTART ( ..., _rc=myrc) ;
...
```

%CPDUPCHK

Checks for duplicate data by examining timestamps on data that is processed into the PDB

%CPDUPCHK Overview

If you use the %CMPROCES or %CPPROCES macro to process data and enable duplicate data checking (by setting the DUPMODE= parameter to FORCE, DISCARD, or TERMINATE), then the %CPDUPCHK macro checks the incoming data for duplicates. Based on the data's timestamp (which indicates when the input data was created by the data source or data collector) and the value of the IDVAR= parameter (which indicates the originator of the data, such as machine or system), the %CPDUPCHK macro prevents duplicate data from being processed into the active PDB or allows "duplicate" data to be processed into the active PDB, depending on the settings of the FORCE= and TERM= parameters.

When you modify your code to support duplicate-data checking, SAS data sets are created by the duplicate-data-checking macros. These data sets are used to store the timestamp, originator, and collector information for the data that is being checked. These data sets are referred to as temporary, intermediate, and permanent control data sets.

If you specify that you do not want duplicate data to be processed into the PDB, then %CPDUPCHK will read the timestamp and originator of the incoming data and then reject the data if the timestamp for the data falls within an already-processed datetime range for the originator (as recorded in the permanent control data set that corresponds to the collector or data source).

If you specify that you want to allow duplicate data to be processed into the active PDB, then %CPDUPCHK will read the timestamp and originator of the incoming data. However, even if the timestamp falls within an already-processed datetime range for the originator (as recorded in the permanent control data set that corresponds to the collector or data source), the data will still be processed into the active PDB. (The "duplicate" data should be data that may appear to the duplicate-data-checking macros to be duplicate, but is not actually duplicate. For example, the "duplicate" data was processed into the PDB before but was accidentally deleted. Or the "duplicate" data is being processed into a table that was not in use in the PDB when the data was processed earlier.)

For more information about duplicate-data checking, see "Duplicate-Data Checking" on page 671 and, near the end of that topic, the case that applies to your collector or data source.

%CPDUPCHK Syntax

%CPDUPCHK(
ENDFILE=*variable-name*
,**IDVAR=***variable-name*
,**SOURCE=***identifier*
,**TIMESTMP=***timestamp-variable-name*
<,**FORCE=**YES | NO>
<,**INT=***interval*>
<,**KEEP=***number-of-weeks*>
<,**RANGES=***number-of-ranges*>
<,**SYSTEMS=***number-of-systems*>
<,**TERM=**YES | NO>);

Details

ENDFILE=*variable-name*
specifies the name of the observation's variable that is specified by the END= parameter on the SAS INFILE or SET statement that reads your raw data. *This parameter is required.*

In either of the following cases, you must provide the value of ENDFILE=.

- You use COLLECTR=GENERIC in your call to %CMPROCES or %CPPROCES.
- You have created a set of collector-support entities for installation and use with %CPPROCES.

Otherwise, the supplied collector support provides the appropriate value from this table.

Table 2.3 Values for the ENDFILE= Parameter

COLLECTR=	Process Macro	ENDFILE=
ACCUNX	%CP	_LAST
DCOLLECT	%CM	EOF
EREP	%CM	ENDOF
ETEWATCH	%CP	DUPEOF
GENERIC	%CM	See Notes
GENERIC	%CP	See Notes
HP-PCS	%CP	LAST
IMF	%CM	END
NETCONV	%CP	_EOF
NTSMF	%CP	_EOF
NTSMF	%CM	END

COLLECTR=	Process Macro	ENDFILE=
PATROL	%CP	LAST
ROLMPBX	%CP	See Notes
SAPR3	%CP	DUPEND
SAR	%CP	_ENDSAR
SITESCOP	%CP	EOF
SMF	%CM	ENDOFSMF
SMF	%CP	ENDOFSMF
TMONCICS	%CM	LASTLINE
TMONCIC8	%CM	END
TMONDB2	%CM	END
TMON2CIC	%CM	LASTLINE
TMS	%CM	TMSEND
TPF	%CM	ENDOFTPF
VISULIZR	%CP	LAST
VMMON	%CM	ENDOFILE
WEBLOG	%CP	DUPEOF

Notes:

1. The COLLECTR= column contains the value of the COLLECTR= parameter.
2. The Process Macro column contains the first two letters in the name of the macro that is used to process the data: %CM for %CMPROCES, and %CP for %CPPROCES.
3. The ENDFILE= column contains the value to use for this parameter, if you are using the corresponding collector and process macro.
4. For COLLECTR=NTSMF, there are two sets of values. The first is for %CPPROCES using COLLECTR=NTSMF and TOOLNM=SASDS. The second is for %CMPROCES using COLLECTR=NTSMF and TOOLNM=MXG.
5. For COLLECTR=ROLMPBX, duplicate-data checking has not been implemented.
6. For COLLECTR=GENERIC: For the value to use for ENDFILE=, look in your staging code.

   The %CPDUPCHK macro monitors information such as system names, timestamp ranges, and record counts while the data is being processed. When the last record is processed, all of this information is written to a temporary control data set and then passed to the %CPDUPUPD macro. The %CPDUPUPD macro writes to one or more intermediate data sets and uses them to update one or more of the permanent control data sets.

   The %CPDUPCHK macro depends on being signaled by an end-of-file condition. The INFILE statement that references the external data source must contain the END= parameter, because the value of this parameter is the name of the SAS variable that %CPUDPCHK uses in order to check for the end-of-file condition.

   If the END= parameter is not specified, or if its value does not match the value of the ENDFILE= parameter on the %CPDUPCHK macro that is

specified in the next section, then the control data cannot be written to the temporary control data set and you will receive an ERROR message. Correct the error (as described in the next paragraph) and confirm that the value of the END= parameter on the INFILE statement matches the value of the ENDFILE= parameter on the %CPDUPCHK macro. Then, reprocess the data. Because the timestamps for the data were not recorded, you can reprocess the data as if it were new.

To add the END= parameter or modify its value to match the value of the ENDFILE= parameter on %CPDUPCHK, use a statement similar to this one:

```
INFILE INDATA STOPOVER LENGTH=LENGTH
       COL=COL END=xxxxxxxx;
```

In this example, *xxxxxxxx* can be any valid SAS variable name, but it must match the name that is used as the value for the ENDFILE= parameter on the call to the %CPDUPCHK macro for this data. The name that you choose must not match the name of any variables in the staged data set or staged data view.

Note: If you are using MXG code to stage the data:
In MXG code, typically the INFILE statement is in the collector's IMAC or VMAC member or file, but it may be in another of the collector's members or files. If the member or file containing the INFILE statement is not already in MXG.USERID.SOURCLIB, copy that member or file from MXG.MXG.SOURCLIB to MXG.USERID.SOURCLIB. Make modifications, if any, in the copy in MXG.USERID.SOURCLIB. △

IDVAR=*variable-name*
specifies the name of the observation's variable that identifies the originator of the data. Typically, this variable is the one that identifies the machine or system, but this variable can be any variable that uniquely identifies the data. For example, if the system name does not uniquely identify the originator and the host name does not uniquely identify the originator, but the combination of system name and host name does uniquely identify the originator, then you can create a derived variable that concatenates the system name and host name, and use the name of the derived variable as the value of the IDVAR= parameter.

This parameter is required.

In either of the following cases, you must provide the value of IDVAR=:

- You use COLLECTR=GENERIC in your call to %CMPROCES or %CPPROCES.
- You have created a set of collector-support entities for installation and use with %CPPROCES.

Otherwise, the supplied collector support provides the appropriate value from this table.

Table 2.4 Values for the IDVAR= Parameter

COLLECTR=	Process Macro	IDVAR=
ACCUNX	%CP	MACHINE
DCOLLECT	%CM	DCUSYSID
EREP	%CM	CPUSER
ETEWATCH	%CP	MACHINE

COLLECTR=	Process Macro	IDVAR=
GENERIC	%CM	See Notes
GENERIC	%CP	See Notes
HP-PCS	%CP	MACHINE
IMF	%CM	SYSTEM
NETCONV	%CP	MACHINE
NTSMF	%CP	MACHINE
NTSMF	%CM	SYSTEM
PATROL	%CP	MACHINE
ROLMPBX	%CP	See Notes
SAPR3	%CP	SYSHOST
SAR	%CP	MACHINE
SITESCOP	%CP	MONITOR
SMF	%CM	SYSTEM
SMF	%CP	SYSTEM
TMONCICS	%CM	SYSTEM
TMONCIC8	%CM	SYSTEM
TMONDB2	%CM	SSSID
TMON2CIC	%CM	SYSTEM
TMS	%CM	TMCNAME
TPF	%CM	CPUID
VISULIZR	%CP	MACHINE
VMMON	%CM	VMSYSTEM
WEBLOG	%CP	MACHINE

Notes:

1. The COLLECTR= column contains the value of the COLLECTR= parameter.
2. The Process Macro column contains the first two letters in the name of the macro that is used to process the data: %CM for %CMPROCES, and %CP for %CPPROCES.
3. The IDVAR= column contains the value that should be used for this parameter, if you are using the corresponding collector and process macro.
4. For COLLECTR=NTSMF, there are two sets of values. The first is for %CPPROCES using COLLECTR=NTSMF and TOOLNM=SASDS. The second is for %CMPROCES using COLLECTR=NTSMF and TOOLNM=MXG.
5. For COLLECTR=ROLMPBX, duplicate-data checking has not been implemented.
6. For COLLECTR=GENERIC, use the name of the variable whose value uniquely identifies the originator of the data.

SOURCE=*identifier*
specifies a three-character identifier of your data collector or data source.
This parameter is required.
In either of the following cases, you must provide the value of SOURCE=:

- You use COLLECTR=GENERIC in your call to %CMPROCES or %CPPROCES.
- You have created a set of collector-support entities for installation and use with %CPPROCES.

Otherwise, the supplied collector support provides the appropriate value from this table.

For more information related to the SOURCE= parameter, see "Duplicate-Data Checking" on page 671.

Table 2.5 Values for the SOURCE= Parameter

COLLECTR=	Process Macro	SOURCE=
ACCUNX	%CP	ACC
DCOLLECT	%CM	DCO
EREP	%CM	ERP
ETEWATCH	%CP	ETE
GENERIC	%CM	See Notes
GENERIC	%CP	See Notes
HP-PCS	%CP	See Notes
IMF	%CM	IMF
NETCONV	%CP	NET
NTSMF	%CP	NTS
NTSMF	%CM	NTS
PATROL	%CP	PAT
ROLMPBX	%CP	See Notes
SAPR3	%CP	See Notes
SAR	%CP	SAR
SITESCOP	%CP	STS
SMF	%CM	SMF
SMF	%CP	SMF
TMONCICS	%CM	TMC
TMONCIC8	%CM	TM8
TMONDB2	%CM	TMD
TMON2CIC	%CM	TM2
TMS	%CM	TMS
TPF	%CM	TPF
VISULIZR	%CP	See Notes
VMMON	%CM	VMM
WEBLOG	%CP	WWW

Notes:

1. The COLLECTR= column contains the value of the COLLECTR= parameter.
2. The Process Macro column contains the first two letters in the name of the macro that is used to process the data: %CM for %CMPROCES, and %CP for %CPPROCES.
3. The SOURCE= column contains the value that should be used for this parameter if you are using the corresponding collector and process macro.
4. For COLLECTR=GENERIC, the identifier must start with a letter and can contain both letters and numbers. The identifier should be different from any of the identifiers in this table.
5. For COLLECTR=HP-PCS, the value of SOURCE= is the 4th, 5th, and 6th characters in the table's name. For example, if the table's name is PCSABC, then SOURCE=ABC.
6. For COLLECTR=NTSMF, there are two sets of values. The first is for %CPPROCES using COLLECTR=NTSMF and TOOLNM=SASDS. The second is for %CMPROCES using COLLECTR=NTSMF and TOOLNM=MXG.
7. For COLLECTR=SAPR3, there are three sets of values. For table SAPR3S, use SOURCE=SAP. For table SAPBTCH, use SOURCE=BAT. For the remaining tables, the value of SOURCE= is the 4th, 5th, and 6th characters in the table's name. For example, if the table's name is SAPABC, then SOURCE=ABC.
8. For COLLECTR=ROLMPBX, duplicate-data checking has not been implemented.
9. For COLLECTR=VISULIZR, the value of SOURCE= is the 4th, 5th, and 6th characters in the table's name. For example, if the table's name is VZWABC, then SOURCE=ABC.

TIMESTMP=*timestamp-variable-name*
specifies the name of the observation's variable that contains the datetime stamp. *This parameter is required.*

In either of the following cases, you must provide the value of TIMESTMP=:

- You use COLLECTR=GENERIC in your call to %CMPROCES or %CPPROCES.
- You have created a set of collector-support entities for installation and use with %CPPROCES.

Otherwise, the supplied collector support provides the appropriate value from this table.

Table 2.6 Values for the TIMESTMP= Parameter

COLLECTR=	Process Macro	TIMESTMP=
ACCUNX	%CP	DATETIME
DCOLLECT	%CM	DCUTMSTP
EREP	%CM	See Notes
ETEWATCH	%CP	DATETIME
GENERIC	%CM	See Notes
GENERIC	%CP	See Notes
HP-PCS	%CP	DATETIME
IMF	%CM	ENDTIME

COLLECTR=	Process Macro	TIMESTMP=
NETCONV	%CP	DATETIME
NTSMF	%CP	DATETIME
NTSMF	%CM	STARTIME
PATROL	%CP	DATETIME
ROLMPBX	%CP	See Notes
SAPR3	%CP	DATETIME
SAR	%CP	DATETIME
SITESCOP	%CP	DATETIME
SMF	%CM	SMFTIME
SMF	%CP	SMFTIME
TMONCICS	%CM	TMMDCCLK
TMONCIC8	%CM	TMMDCCLK
TMONDB2	%CM	REGNTIME
TMON2CIC	%CM	TMMDCCLK
TMS	%CM	DATETIME
TPF	%CM	TPFTIME
VISULIZR	%CP	DATETIME
VMMON	%CM	MRHDRTOD
WEBLOG	%CP	DATETIME

Notes:

1. The COLLECTR= column contains the value of the COLLECTR= parameter.
2. The Process Macro column contains the first two letters in the name of the macro that is used to process the data: %CM for %CMPROCES, and %CP for %CPPROCES.
3. For COLLECTR=EREP, there are three values for the TIMESTMP= parameter: INCDTIME is the default value; DATETIME is the value for the XERPTIM-Timestamp Record; INITTIME is the value for the XERPIPL-IPL Record.
4. For COLLECTR=GENERIC, use the variable that you have designated to be the DATETIME variable. If, however, there are two datetime variables and the other variable has the datetime stamp for when the record was created, use that other variable instead of the variable that you have designated to be the DATETIME variable.
5. For COLLECTR=NTSMF, there are two sets of values. The first is for %CPPROCES using COLLECTR=NTSMF and TOOLNM=SASDS. The second is for %CMPROCES using COLLECTR=NTSMF and TOOLNM=MXG.
6. For COLLECTR=ROLMPBX, duplicate-data checking has not been implemented.

FORCE=YES | NO
specifies whether or not duplicate data is to be processed into the PDB. FORCE=YES indicates that duplicate data should be processed into the PDB. FORCE=NO indicates that duplicate data should not be processed into the PDB. *The default value is NO*.

%CPPROCES and %CMPROCES check whether FORCE is requested by either DUPMODE=FORCE (on the %CPPROCES or %CMPROCES call) or FORCE=YES on the %CPDUPCHK call or both. If either or both are true, the FORCE option in processing is in effect.

In some cases, you do use FORCE=YES on the %CPDUPCHK macro when DUPMODE=FORCE on the call to the %CMPROCES or %CPPROCES macro. In other cases, you do not use FORCE=YES on the %CPDUPCHK macro when DUPMODE=FORCE on the call to the %CMPROCES or %CPPROCES macro. The relation between the FORCE= parameter and the DUPMODE= parameter depends on which duplicate-data-checking case applies to your data. For a more complete description, for implementation instructions, and for examples, see "Duplicate-Data Checking" on page 671 and, near the end of that topic, see the case that applies to your collector or data source.

Remember to change the DUPMODE= and/or FORCE= settings to their earlier values after you finish using them to allow "duplicate" data into the PDB.

INT=*interval*

represents the maximum interval that is to be allowed between the timestamps on any two consecutive records from the same system or machine. If the interval between the timestamp values exceeds the value of this parameter, then a new time range is created in the temporary control data set. This is referred to as a gap in the data. *The default value for this parameter is 0:29, or 29 minutes.* Review your collector's data before using this default value or setting a different value.

If a gap is detected, then you will receive a warning message in the SAS log and a report is printed to standard SAS output. The report lists the timestamps for the data that caused the error, as well as all the stored timestamps for that IDVAR= value, which is typically the name of the machine or system.

The value for this parameter must be provided in the format *hh:mm:ss*, where *hh* is hours, *mm* is minutes, and *ss* is seconds. For example, to specify an interval of 14 minutes, use INT=0:14. To specify an interval of 1 hour and 29 minutes, use INT=1:29. (See also the RANGES= parameter.)

KEEP=*number-of-weeks*

represents the maximum number of weeks for which you want to retain control data in the PDB. The control data sets store a permanent record of each collector's DATETIME ranges for each value of the parameter that is specified by the IDVAR= parameter. When you set the KEEP= parameter, a range's control data is aged out or removed when the datetime value is older than the number of weeks that is specified by the value of this parameter. *The default value is 9.*

When the permanent control data sets are updated by the %CPDUPUPD macro, the end-of-range timestamp of each range is compared with the timestamp implied by the value of the KEEP= parameter. If the end-of-range datetime value is older than the timestamp that is implied by the KEEP= parameter, then the data is removed.

If your data is continuous, then you will have only one range and your control information will never age out, because the end-of-range datetime value is constantly extended by new datetime information.

RANGES=*number-of-ranges*

represents the maximum number of ranges that can occur during the current execution of %CMPROCES or %CPPROCES. A new range is created when the difference between the datetime stamps of two consecutive observations for the same value of the variable that is specified by the IDVAR= parameter exceeds the value of the INT= parameter. *The default is 20 and the maximum is 999*.

When the difference between two consecutive datetime stamps exceeds the value of the INT= parameter, the interval is referred to as a gap in the data. Gaps

in the data should be considered an exceptional event, but you might want to overestimate the RANGE= value by a small amount to prevent needless reruns due to error messages. Do not overestimate by a large amount, because an extremely large value can affect memory utilization.

SYSTEMS=*number-of-systems*
specifies the maximum number of values of the variable that is specified by the IDVAR= parameter for which you expect to process data. *The default is three and the maximum is 99,999*.

If the number of values found in the data exceeds the value of this parameter, then you will receive ERROR messages that indicate an insufficient number of memory slots. Therefore, you may want to overestimate the current maximum by a small amount to allow for growth. Do not overestimate by a large amount, because an extremely large value can affect memory utilization.

TERM=YES | NO
specifies whether or not to terminate with an error condition if duplicate data is found in the data that you are processing.

- TERM=YES indicates that termination is to occur when duplicate data is found.

  In interactive mode, if TERM=YES and duplicate data is detected, then

  - the process step terminates
  - an ERROR message displays at the end of the SAS log
  - the PDB is not deactivated
  - the SAS IT Resource Management session continues
  - the underlying SAS session continues.

  In batch mode, if TERM=YES and duplicate data is detected, the job abends.
- TERM=NO indicates that termination is not to occur when duplicate data is found.

The default value is NO.

%CPPROCES and %CMPROCES check whether TERMINATE is requested by either DUPMODE=TERMINATE (on the %CPPROCES or %CMPROCES call) or TERM=YES on the %CPDUPCHK call or both. If either or both are true, the TERMINATE option in processing is in effect.

In some cases you do use TERM=YES on the %CPDUPCHK macro when DUPMODE=TERMINATE on the call to the %CMPROCES or %CPPROCES macro. In other cases, you do not use TERM=YES on the %CPDUPCHK macro when DUPMODE=TERMINATE on the call to the %CMPROCES or %CPPROCES macro. The relation between the TERM= parameter and the DUPMODE= parameter depends on which duplicate-data-checking case applies to your data. For a more complete description, for implementation instructions, and for examples, see “Duplicate-Data Checking” on page 671 and, near the end of that topic, see the case that applies to your collector or data source.

%CPDUPCHK Notes

For a more complete description, for implementation instructions, and for examples, see “Duplicate-Data Checking” on page 671 and, near the end of that topic, the case that applies to your collector or data source.

%CPDUPDSN

Generates the name of the temporary data set that will contain datetime ranges for the data that is being processed into the active PDB

%CPDUPDSN Overview

The %CPDUPDSN macro creates the name of the SAS data set that will contain the datetime information for data that is being processed into the active PDB, by %CMPROCES or %CPPROCES, when duplicate-data checking is enabled. When you process data and use duplicate-data checking, the datetime information for the data is saved in the data set that is created by %CPDUPDSN, and this information is used by other duplicate-data-checking macros, such as %CPDUPCHK and %CPDUPUPD.

If you are using supplied collector-support software, then the %CPDUPDSN macro is automatically submitted for you when you turn on duplicate-data checking by using the DUPMODE= parameter on %CPPROCES or %CMPROCES.

If you are using the Generic Collector Facility, then you must enable duplicate-data checking by specifying the DUPMODE= parameter and by calling %CPDUPDSN from your staging code.

%CPDUPDSN Syntax

%CPDUPDSN(
SOURCE=*collector-id*);

Details

SOURCE=*identifier*

specifies a three-character identifier of your data collector or data source.

This parameter is required.

In either of the following cases, you must provide the value of SOURCE=:

- You use COLLECTR=GENERIC in your call to %CMPROCES or %CPPROCES.
- You have created a set of collector-support entities for installation and use with %CPPROCES.

Otherwise, the supplied collector support provides the appropriate value from this table.

For more information related to the SOURCE= parameter, see “Duplicate-Data Checking” on page 671.

Table 2.7 Values for the SOURCE= Parameter

COLLECTR=	Process Macro	SOURCE=
ACCUNX	%CP	ACC
DCOLLECT	%CM	DCO
EREP	%CM	ERP
ETEWATCH	%CP	ETE

COLLECTR=	Process Macro	SOURCE=
GENERIC	%CM	See Notes
GENERIC	%CP	See Notes
HP-PCS	%CP	See Notes
IMF	%CM	IMF
NETCONV	%CP	NET
NTSMF	%CP	NTS
NTSMF	%CM	NTS
PATROL	%CP	PAT
ROLMPBX	%CP	See Notes
SAPR3	%CP	See Notes
SAR	%CP	SAR
SITESCOP	%CP	STS
SMF	%CM	SMF
SMF	%CP	SMF
TMONCICS	%CM	TMC
TMONCIC8	%CM	TM8
TMONDB2	%CM	TMD
TMON2CIC	%CM	TM2
TMS	%CM	TMS
TPF	%CM	TPF
VISULIZR	%CP	See Notes
VMMON	%CM	VMM
WEBLOG	%CP	WWW

Notes:

1. The COLLECTR= column contains the value of the COLLECTR= parameter.
2. The Process Macro column contains the first two letters in the name of the macro that is used to process the data: %CM for %CMPROCES, and %CP for %CPPROCES.
3. The SOURCE= column contains the value that should be used for this parameter if you are using the corresponding collector and process macro.
4. For COLLECTR=GENERIC, the identifier must start with a letter and can contain both letters and numbers. The identifier should be different from any of the identifiers in this table.
5. For COLLECTR=HP-PCS, the value of SOURCE= is the 4th, 5th, and 6th characters in the table's name. For example, if the table's name is PCSABC, then SOURCE=ABC.
6. For COLLECTR=NTSMF, there are two sets of values. The first is for %CPPROCES using COLLECTR=NTSMF and TOOLNM=SASDS. The second is for %CMPROCES using COLLECTR=NTSMF and TOOLNM=MXG.
7. For COLLECTR=SAPR3, there are three sets of values. For table SAPR3S, use SOURCE=SAP. For table SAPBTCH, use SOURCE=BAT. For the

remaining tables, the value of SOURCE= is the 4th, 5th, and 6th characters in the table's name. For example, if the table's name is SAPABC, then SOURCE=ABC.

8 For COLLECTR=ROLMPBX, duplicate-data checking has not been implemented.

9 For COLLECTR=VISULIZR, the value of SOURCE= is the 4th, 5th, and 6th characters in the table's name. For example, if the table's name is VZWABC, then SOURCE=ABC.

%CPDUPDSN Notes

Currently, this macro and other duplicate data checking macros are available only if you are processing data by using %CPPROCES or %CMPROCES.

For complete details on implementing this macro, see "Duplicate-Data Checking" on page 671.

%CPDUPINT

Loads macro definitions that are used by other duplicate-data-checking macros

%CPDUPINT Overview

%CPDUPINT loads macro definitions that are required in order to check for duplicate data. If you are using supplied collector-support software, then this macro is executed for you. However, if you are using the Generic Collector Facility, then you must insert the %CPDUPINT macro in your staging code.

%CPDUPINT Syntax

%CPDUPINT;

%CPDUPINT Notes

This macro takes no parameters.

For instructions on how and where to insert this macro, refer to the implementation section that is related to your setup method and collector in "Duplicate-Data Checking" on page 671.

%CPDUPUPD

Updates the datetime information in the control data sets for duplicate-data checking

%CPDUPUPD Overview

When you enable duplicate-data checking for data that is processed with the %CMPROCES and %CPPROCES macros, the %CPDUPUPD macro is submitted by the

%CMPROCES or %CPPROCES macro to update the permanent control data sets after %CPDUPCHK checks the data. %CPDUPUPD serves as a part of the set of duplicate data checking macros, but you do not call the %CPDUPUPD macro directly.

When new data is processed, the %CPDUPCHK macro creates a temporary control data set that is named WORK._DUPCNTL in which to store information about the input data. After the data has been checked by %CPDUPCHK, the %CPDUPUPD macro reads WORK._DUPCNTL and divides the information into separate, intermediate control data sets (one data set for each collector). These intermediate control data sets are named ADMIN.*source*DAY, where *source* is the three-character identifier of the data collector or data source that was specified using the SOURCE= parameter on the %CPDUPCHK macro.

%CPDUPUPD then merges the new datetime information, which is in the ADMIN.sourceDAY files, with the existing datetime information in the permanent control data sets, named ADMIN.*source*CNTRL, where again *source* represents the three-character identifier of the data collector or data source that was specified by using the SOURCE= parameter on the %CPDUPCHK macro.

%CPDUPUPD Syntax

%CPDUPUPD;

%CPDUPUPD Notes

This macro takes no parameters.

For more information about duplicate-data checking when you are processing data with the %CMPROCES or %CPPROCES macro, see “Duplicate-Data Checking” on page 671.

%CPEDIT

Edits or deletes data in the active PDB

%CPEDIT Overview

The %CPEDIT macro enables you to edit or delete data in a PDB. This macro maintains the integrity of the data dictionary while it performs this activity. Because of the risks that are associated with incorrectly editing a PDB, you can edit only a single table at a single PDB level, each time you invoke this macro.

Note: You should consult SAS Technical Support before using this macro in order to preserve the integrity of the PDB. △

%CPEDIT is useful in situations where erroneous data must be either removed from the PDB or corrected in the PDB. You might also use %CPEDIT to alter the name of a machine that has changed. You can edit or delete data by specifying one of the following parameters: KEEPCRIT=, DELCRIT=, or SRCFILE=. If your data manipulation needs are highly complex, then edit your data in phases by submitting a separate call to the macro for each of these parameters.

By default, the %CPEDIT macro does not incorporate your changes if those changes would result in zero observations in the specified level of the specified table. You can override this default by using the ZOBSOK= parameter.

%CPEDIT Syntax

%CPEDIT(
TABLE=*table-name*
,**LEVEL=**DETAIL | DAY | WEEK | MONTH | YEAR
,**KEEPCRIT=***keep-criteria* | **DELCRIT=***delete-criteria* |
SRCFILE=*ext-file-name*
<,**_RC=***macro-var-name*>
<,**ZOBSOK=**Y | N>);

Details

TABLE=*table-name*
specifies the name of the table whose data you want to edit in the active PDB. The table must already exist in the active PDB.

The modifications apply to the data in the level (specified by the LEVEL= parameter) of the table that is specified by the TABLE= parameter.

LEVEL=DETAIL | DAY | WEEK | MONTH | YEAR
specifies the level of the PDB where you want to modify your data. The modifications apply to the data in the level (specified by the LEVEL= parameter) of the table that is specified by the TABLE= parameter.

KEEPCRIT=*keep-criteria*
specifies the criteria to use for keeping data in the specified level of the specified table. The value that you supply for this parameter should be written as a Boolean expression by using the guidelines for writing expressions in SAS WHERE clauses.

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. Do not use any formula variables in the criteria for keeping data. △

Observations are kept in the table if they meet the criteria that are specified in this expression, such that the expression is True.

You must specify one of the following parameters: SRCFILE=, DELCRIT=, or KEEPCRIT=. If you specify this parameter, then you cannot specify the DELCRIT= or SRCFILE= parameter.

DELCRIT=*delete-criteria*
specifies the criteria to use for deleting data from the specified level of the specified table. The value that you supply for this parameter should be written as a Boolean expression by using the guidelines for writing expressions in SAS WHERE clauses.

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. Do not use any formula variables in the criteria for deleting data. △

Observations are deleted from the PDB if they meet the criteria that are specified in this expression, such that the expression is True.

You must specify one of the following parameters: SRCFILE=, DELCRIT=, or KEEPCRIT=. If you specify this parameter, then you cannot specify the KEEPCRIT= or SRCFILE= parameter.

SRCFILE=*ext-file-name*
specifies the complete path and name of an external file that contains the SAS DATA step code that is used to perform your data modifications.

The code in the external file that is specified for this parameter should not contain any of the following SAS statements: DROP, KEEP, INCLUDE, RENAME, DELETE, SET, OUTPUT, STOP, ABORT, MERGE, MODIFY, BY, LENGTH, FORMAT, ATTRIB, DATA, or RUN.

Do not create any variables in any of the SAS statements in the SRCFILE= parameter file.

You must specify one of the following parameters: SRCFILE=, DELCRIT=, or KEEPCRIT=. If you specify this parameter, then you cannot specify the KEEPCRIT= or DELCRIT= parameters.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

ZOBSOK=Y | N
specifies whether your data modifications should be made if the specified level of the specified table would have zero observations when the actions specified in the DELCRIT=, KEEPCRIT=, or SRCFILE= parameter were taken. The value can be Y (meaning YES) or N (meaning NO). *The default value is N.*

This parameter is provided as insurance if you want the DELCRIT=, KEEPCRIT=, or SRCFILE= parameter to delete only a subset of the observations. If a problem with the syntax of one of these parameters would cause all the observations to be deleted, then the default value of N prevents this from happening.

%CPEDIT Notes

Always make a backup of your PDB before you run this macro.

This macro works on only one table at a time, and at only one level of the PDB for that table. In order to edit all levels of the PDB for a given table, you must submit the macro separately for each level in the table.

This macro attempts to perform limited syntax checking of the KEEPCRIT= and SRCFILE= parameters as well as syntax checking of the statements that are contained in the file that SRCFILE= points to. However, this syntax checking is unable to protect against such errors as misspelled or missing variable names.

Do not create any variables in any of the SAS statements in the SRCFILE= parameter file.

Do not use this macro if you are having any problems with your PDB, such as an unrecovered checkpoint in reduction. Also, do not use this macro between the time that you process your data and the time that you reduce that same data.

%CPEDIT Examples

Example 1

Keep all observations in the DAY level that are later than 01Jan2003 for table XTY70. This example does not commit the changes if the specified level of the specified table has zero observations.

```
%cpedit(table=xty70,level=day,
   keepcrit=DATEPART(DATETIME)>'01Jan2003'd,
   zobsok=N);
```

Example 2

Delete any observations for machine ABC1 in the WEEK level for table XTY70. Commit the changes, even if the specified level of the specified table has zero observations.

```
%cpedit(table=xty70,level=week,delcrit=MACHINE='ABC1',zobsok=Y);
```

Example 3

Edit the data, such that any machines that are called XYZ2 are now called ABC1 in the YEAR level for table XTY70.

This example uses an external file that is named T.T, which contains the following SAS DATA step statements:

```
if MACHINE='XYZ2' then MACHINE='ABC1';
```

The %CPEDIT macro is invoked as follows:

```
%cpedit(table=xty70,level=year,srcfile=T.T);
```

%CPEISSUM

Prepares and outputs PDB data

%CPEISSUM Overview

The %CPEISSUM macro is used to prepare data (from one or more views in the active PDB) for reports. Data preparation has five steps: selection, subsetting, transformation, summarization, and output.

The selection step identifies the views (level_name.table_name combinations) whose data is to be summarized. The subsetting step identifies what subset of the views' data is to be used. In both the macro's auto-summarizing and self-summarizing modes, you control both these steps by specifying parameters. When the macro runs, it generates and submits code that corresponds to the values of these parameters.

The transformation step identifies any additional transformations that the data is to undergo. The summarization step summarizes the data. In the macro's auto-summarizing mode, you control both these steps by specifying parameters. When the macro runs, it generates and submits code that corresponds to the values of these parameters. In the macro's self-summarizing mode, you control both these steps by specifying a parameter that points to prewritten code, which the macro submits in order to implement these steps.

The output step identifies the location to which the results are to be written, in both the macro's auto-summarizing and self-summarizing modes. This is not always performed as a separate step, because the summarization step usually writes the results to the output locations.

For auto-summarizing mode, specify parameters for this macro as follows:

- Required parameters: tablist; OUTLIB=; at least one of the N_level parameters; either CLASS= (if SUMPROC= is set to SUMMARY) or both CLASS= and HIERS= (if SUMPROC= is set to or defaults to MDDB); and VAR=.
- Optional parameters: the remaining N_level parameters; WHERE; OUTMEMS= and OUTLABS=; TOPN=, TOPNCLAS=, and TOPNVAR=; STACKTYP=, STACKVAL=, and STACKVAR=; SUMPROC= and STATS=.
- If you specify the VAR= parameter, then this macro executes in auto-summarizing mode and the following parameters are not used: SOURCAT= and HIERS= (if SUMPROC= is set to or defaults to SUMMARY).

For self-summarizing mode, use the parameters as follows:

- Required parameters: tablist; OUTLIB=; and at least one of the N_level parameters; you must not use the VAR= parameter.
- Optional parameters: the remaining N_level parameters; WHERE=; OUTMEMS= and OUTLABS=; and SOURCAT=.
- If you do not specify the VAR= parameter, then the %CPEISSUM macro executes in self-summarizing mode and the following parameters are not used: TOPN=, TOPNCLAS=, and TOPNVAR=; STACKTYP=, STACKVAL=, and STACKVAR=; SUMPROC=, CLASS=, HIERS=, and STATS=.

%CPEISSUM Syntax

%CPEISSUM(
tablist
,**OUTLIB=***libref*
<,**CLASS=**>*variable-list*
<,**HIERS=***variable-list*>
<,**N_DAYS=***number-of-days*>
<,**N_DETAIL=***number-of-days*>
<,**N_MONTHS=***number-of-months*>
<,**N_WEEKS=***number-of-weeks*>
<,**N_YEARS=***number-of-years*>
<,**OUTLABS=***label-list*>
<,**OUTMEMS=***module-names*>
<,**_RC=***macro-var-name*>
<,**SOURCAT=***libref*>
<,**STACKTYP=***variable-name*>
<,**STACKVAL=***variable-name*>
<,**STACKVAR=**NO | YES>
<,**STATS=***statistics-list*>
<,**SUMPROC=**MDDB | SUMMARY>
<,**TOPN=***integer*>
<,**TOPNCLAS=***variable-list*>
<,**TOPNVAR=***variable-name*>
<,**VAR=**>*variable-list*
<,**WHERE=***where-expression*>);

Details

tablist
lists the names of one or more tables whose data is to be selected, subset, transformed, summarized, and output, potentially for use with the SAS IT Resource Management DeskTop Reporter. If you list more than one table name, then separate the table names with one or more blanks.

In both self-summarizing and auto-summarizing mode, this parameter is required.

For more information about the views that are selected from these tables, see the N_*level* set of parameters.

OUTLIB=*libref*
specifies the name of a libref to which results are written.

In auto-summarizing mode, the libref should represent the final library to which the results are to be written. Use DOWNLOAD for this libref.

In self-summarizing mode, the libref should represent the library to which the intermediate results are to be written (the results from the DATA step that %CPEISSUM generates, based on the selection and subsetting criteria). Typically, WORK is used for this libref.

In both self-summarizing and auto-summarizing mode, this parameter is required.

CLASS=*variable-list*
specifies a list of class variables. If you list more than one variable name, then separate the variable names with one or more blanks.

This parameter is required only if you are using auto-summarizing mode. It is ignored for self-summarizing mode.

In auto-summarizing mode, if the SUMPROC= parameter is set to (or defaults to) MDDB, then this parameter specifies the list of class variables that are to be used in PROC MDDB in order to group the data.

In auto-summarizing mode, if the SUMPROC= parameter is set to SUMMARY, then this is the list of class variables that are to be used in PROC SUMMARY in order to group the data.

A variable that is in the CLASS= variables list must not be in the VAR= variables list, and it must exist in every view that is specified by the *tablist* and N_*level* set of parameters. For more information about the class variables, see the SUMMARY procedure in the SAS/MDDB documentation.

HIERS=*variable-list*
specifies one or more hierarchy lists. Each hierarchy list consists of one or more variable names from the list that is specified by the CLASS= parameter and becomes a named hierarchy. If there is more than one name in a hierarchy list, then separate the names with one or more spaces. If there is more than one hierarchy list, then separate the lists with an exclamation point (!).

For example, if you specify HIERS=MACHINE DATE, then the following HIERARCHY statement is generated for the PROC MDDB step:

```
HIERARCHY Machine Date / name=MachineDate display=yes;
```

In auto-summarizing mode, if you specify SUMPROC=MDDB, then this parameter is required; if you specify SUMPROC=SUMMARY, then this parameter is not used. In self-summarizing mode, this parameter is not used.

The name of the hierarchy is a concatenation of the names in the hierarchy list.

For more information about the hierarchical lists, see the MDDB procedure in the *SAS Procedures Guide*.

N_DETAIL, N_DAYS, N_WEEKS, N_MONTHS, N_YEARS=
specifies the number of units of data to summarize from each table that is named in the *tablist* parameter. Specify the number of units, but do not specify the unit, such as day, week, month, or year. The units are implied: days at the DETAIL and DAY levels, weeks at the WEEK level, months at the MONTH level, and years at the YEAR level.

In auto-summarizing mode and self-summarizing mode, at least one of these parameters is required. The default value for all levels is zero.

Each N_*level* parameter that has a nonzero value indicates that data is to be prepared at its level from the tables that are listed in the tablist parameter. For example, suppose the tablist parameter is set to XTY70 XTY71 and you specify N_DETAIL=5, N_DAYS=10, and N_MONTHS= 2 (thus N_WEEKS=0 and N_YEARS=0). This indicates that data is to be selected and subset as follows from the active PDB: detail.xty70 (most recent 5 days), detail.xty71 (most recent 5 days), day.xty70 (most recent 10 days), day.xty71 (most recent 10 days), months.xty70 (most recent 2 months), and months.xty71 (most recent 2 months).

For additional subsetting that applies to these subsets of these views, see the WHERE= parameter.

OUTLABS=*label-list*
specifies a list of labels that are to be associated with the modules that are listed in the OUTMEMS= parameter. The maximum length of a label is 40 characters. Wrap double quotation marks around a label that contains a single quotation mark. Otherwise, wrap single quotation marks around a label. If there is more than one label in the list, then separate the labels with an exclamation point (!).

In auto-summarizing and self-summarizing mode, this parameter is optional and the default values for the labels are blank.

If you use the OUTLABS= parameter, then there must be as many labels as there are tables in the *tablist* parameter. Each label in the OUTLABS= list corresponds to the table that is in the same position in the *tablist*. (The first label corresponds to the first table, the second label corresponds to the second table, and so on.)

In auto-summarizing mode, these labels apply to the modules that contain the final results. In this context, the word module means data set or MDDB. The modules are data sets if SUMPROC=SUMMARY. The modules are MDDBs if SUMPROC=MDDB.

In self-summarizing mode, these labels apply to the modules that contain the intermediate results. You can carry them forward to the modules that contain the final results if you refer to &outlabs, which is the label that corresponds to the table in the view whose data is being summarized.

OUTMEMS=*module-names*
specifies a list of module names that are to be associated with the intermediate and/or final results. In this context, the word module means data set or MDDB. The module name that you specify can be a maximum length of seven characters. The first character must be a letter; the remaining characters may be letters or numbers. If there is more than one module name in the list, then separate module names with one or more blanks.

To create the module names for these levels (day, week, month, and year), the module names that you specify are concatenated with a letter (or *level*-suffix) to indicate the level of data with which the module is associated. For the detail level this *level*-suffix position is blank. The suffixes for the other levels are: D for data from a view at the day level, W for data from a view at the week level, M for data from a view at the month level, Y for data from a view at the year level.

If you use the OUTMEMS= parameter, then there must be as many module names as there are tables in the tablist parameter. Each module name in the

OUTMEMS= list corresponds to the table that is in the same position in the *tablist*. (The first module name corresponds to the first table, the second module name corresponds to the second table, and so on.) If you do not specify the OUTMEMS= parameter, then the default value of the OUTMEMS= parameter is the value of the *tablist* parameter. That is, the table names are used as the base for module names.

In auto-summarizing mode, the modules are data sets (if SUMPROC=SUMMARY) or MDDBs (if SUMPROC=MDDB) that are written to the final library.

In self-summarizing mode, the modules refer to the intermediate data sets that are created in the DATA step that the %CPEISSUM macro generates. You can carry the names forward to the final modules by using &outmems and the appropriate suffix for the module level (D for day, W for week, etc.).

In self-summarizing mode, the module names with the appropriate *level*-suffix are also used as the names of the catalog entries whose prewritten code is submitted from the catalog that is specified by the SOURCAT= parameter. For more information about the source entries, see the SOURCAT= parameter and also the %CPEISSUM notes for self-summarizing mode.

In both auto-summarizing and self-summarizing mode, this parameter is optional.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

SOURCAT=*libref*
specifies the libref and catalog name of the catalog where the source entries are stored that contain the prewritten code.

While you debug the source entries, you might want to use WORK.TEMP. After you debug the source entries, you might want to use a more permanent library than the one to which the WORK libref points. For example, you might want to use SASUSER, or SITELIB, or the ADMIN library in the active PDB. (Do not use PGMLIB, because the library to which the PGMLIB libref points is replaced when you install SAS IT Resource Management maintenance.)

Reminder: If you use a library whose libref is not automatically defined by the %CPSTART macro, then define the libref before you call %CPEISSUM with this parameter specified. For more information about defining librefs, see the “Notes” section of this macro.

Similarly, the prewritten code must be loaded in the entries in the catalog by the time that %CPEISSUM is called with this parameter specified. For information about creating and loading source entries, see the "Notes" section of this macro.

In auto-summarizing mode, this parameter is not used. In self-summarizing mode, this parameter is optional. The default value is PGMLIB.CPSAMP.

STACKTYP=*variable-name*
specifies the name of a character variable that is to contain the character value that is generated by STACKVAR=YES.

In auto-summarizing mode, if STACKVAR=YES then this parameter is required, and if STACKVAR= NO then this parameter is not used. In self-summarizing mode, this parameter is not used.

The name of the variable can be a maximum of seven characters in length. It can contain only characters and numbers, and the first character must be a letter. The name must not be DURATION, or the same as any of the names in the CLASS= variables list, or the same as any of the names in the VAR= variables list.

For information about the use of this parameter, see the STACKVAR= parameter.

STACKVAL=*variable-name*
specifies the name of a numeric variable that is to contain the numeric value that is generated by STACKVAR= YES.

In auto-summarizing mode, if STACKVAR=YES then this parameter is required, and if STACKVAR= NO then this parameter is not used. In self-summarizing mode, this parameter is not used.

The name of the variable can be a maximum of seven characters in length. It can contain only characters and numbers, and the first character must be a letter. The name must not be DURATION, or the same as any of the names in the CLASS= variables list, or the same as any of the names in the VAR= variables list. For information about the use of this parameter, see the STACKVAR= parameter.

STACKVAR=NO | YES
specifies whether or not you want to transform the data so that you can produce a report in the form of a stacked bar chart or plot.

In auto-summarizing mode, this parameter is optional. *The default value is NO*. In self-summarizing mode, this parameter is not used.

If STACKVAR=YES, then an extra DATA step is generated. The extra DATA step follows the DATA step that is generated based on selection and subsetting criteria. The extra DATA step transforms the data so that by the time the summarization is finished, the results are suitable for stacked bar charts and plots.

If you specify STACKVAR=YES, then

- you must also specify the STACKVAL= parameter and the STACKTYP= parameter.
- the extra DATA step makes a new data set in which each observation in the original data set (the one that is specified by the selection and subsetting criteria) is replaced by a set of *n* observations, where *n* is the number of variables that are specified by the VAR= parameter.

  Each observation in the original data set has the following fields: DURATION, the variables that are specified by the CLASS= parameter, and the variables that are specified by the VAR= parameter.

  Each observation in the new data set has the following fields: DURATION, the variables that are specified by the CLASS= parameter, the variable that is specified by the STACKVAL= parameter, and the variable that is specified by the STACKTYP= parameter.

  Each observation in the new data set corresponds to one variable in the observation from the original data set.

For example, suppose CLASS=cvar1 cvar2, VAR=var1 var2 var3, STACKVAR=YES, STACKVAL=sval, and STACKTYP=styp. In this example, there is an observation in the original data set that has the following data:

```
duration cvar1 cvar2 var1 var2 var3
```

The corresponding observations in the new data set have the following data:

```
duration cvar1 cvar2 sval styp <- where sval=value(var1) and
                                  styp=externalname(var1)
duration cvar1 cvar2 sval styp <- where sval=value(var2) and
                                  styp=externalname(var2)
duration cvar1 cvar2 sval styp <- where sval=value(var3) and
                                  styp=externalname(var3)
```

That is, on the nth observation in the set of new observations, the value of the (new) variable that is named by the STACKVAL= parameter is the old observation's value of the nth variable in the VAR= list. And on the nth observation in the set of new observations, the value of the (new) variable that is named by the STACKTYP= parameter is the data dictionary's value of the external name of the nth variable in the VAR= list. (The value of external name is either the name of the variable in the staged data or, for some data sources, a longer, more descriptive name of the variable.)

STATS=*statistic-list*
lists the statistics to keep for each variable in the VAR= parameter or the STACKVAL= and STACKTYP= parameters. Valid values for this parameter are dependent on the value that you specify for the SUMPROC= parameter.

In auto-summarizing mode, this parameter is optional. The default value is MEAN. In self-summarizing mode, this parameter is not used.

If you specify SUMPROC=SUMMARY, then the list must contain one or more of the following statistic names: MEAN, MIN, and MAX. (The default value is MEAN.) If the list has more than one statistic name, then separate the statistic names with one or more blanks.

On each observation in the resulting data sets, there will be the following variables: DURATION, the variables that are specified by the CLASS= parameter, and variables whose names combine the name of a VAR= variable and a suffix that indicates the statistic name. The suffix for MEAN is blank, the suffix for MINIMUM is N, and the suffix for MAXIMUM is X.

For example, suppose that the incoming analysis variables to PROC SUMMARY are var1 and var2 (either because these are the variables that are identified by the VAR= parameter or because, if STACKVAR=YES, then these are the variables that are identified by the STACKVAL= and STACKTYP= parameters). Suppose also that CLASS=cvar1 cvar2 and STATS= MEAN MAX.

The results will be one observation for all the observations that have a given value of cvar1 and a given value of cvar2. The resulting observation will have this data:

```
duration cvar1 cvar2 var1 var1x var2 var2x
```

In this example,

- the value of var1 is the mean of the var1 values on the observations that were grouped and summarized to make this observation
- the value of var1x is the maximum of the var1 values on the observations that were grouped and summarized to make this observation
- the value of var2 is the mean of the var2 values on the observations that were grouped and summarized to make this observation

- the value of var2x is the maximum of the var2 values on the observations that were grouped and summarized to make this observation.

If you specify (or default to) SUMPROC=MDDB, then the list must contain one or more of the statistic names that are available with the SAS MDDB procedure. (For the complete list of statistic names, see the SAS MDDB procedure in the documentation for your release of the SAS System.)

Note: Some statistics are not on the list because they can be calculated from other statistics that are on the list. For example, the mean can be calculated from the sum and the count. If the list has more than one statistic name, then separate the statistic names with one or more blanks. △

SUMPROC=MDDB | SUMMARY
specifies the name of the procedure to use to summarize the data. Valid values are MDDB or SUMMARY.

In auto-summarizing mode, this parameter is optional. The default value is MDDB. In self-summarizing mode, this parameter is not used.

If you specify SUMPROC=MDDB, then the SAS MDDB procedure (PROC MDDB) is used. If you specify SUMPROC=SUMMARY, then the SAS SUMMARY procedure (PROC SUMMARY) is used.

For more information, see PROC SUMMARY or PROC MDDB in the documentation for your current release of the SAS System.

TOPN=*integer*
specifies an integer. The absolute value of the integer indicates how many groups (N) are to be in the TopN subset.

If the integer is positive, then the macro identifies the N groups that have the highest values of the group property.

If the integer is negative, then the macro identifies the N groups that have the lowest values of the group property.

In auto-summarizing mode, if TOPNVAR= is specified then this parameter is optional and the default value is 10, and if TOPNVAR= is not specified then this parameter is not used. In self-summarizing mode, this parameter is not used.

For information about the use of this parameter, see the TOPNVAR= parameter.

TOPNCLAS=*variable-list*
specifies one or more names of the variable(s) that is to be used to group the observations for TopN calculations. That is, there is to be one group of observations for each unique combination of values of these variables.

In auto-summarizing mode, if TOPNVAR= is specified, then this parameter is optional and the default value is MACHINE, and if TOPNVAR= is not specified, then this parameter is not used. In self-summarizing mode, this parameter is not used.

If you specify STACKVAR=NO (or it defaults to this value), then the variable(s) that you specify in this parameter must be in every view that is specified by the selection and subsetting criteria. The variable(s) can be, but are not required to be, the DURATION variable and/or variables that are specified by the CLASS= or VAR= parameters. If the STACKVAR= parameter is set to YES, then the variable(s) must be from the following list: DURATION, the variables that are specified by the CLASS= parameter, the variable that is specified by the STACKVAL= parameter, and the variable that is specified by the STACKTYP= parameter.

For information about the use of this parameter, see the TOPNVAR= parameter.

TOPNVAR=*variable-name*
name of the variable for which you want the top (or bottom) N number of values.

In auto-summarizing mode, this parameter is optional. If the TOPNVAR= parameter is not specified, then calculations that are defined by the TOPN set of parameters do not take place. If the TOPNVAR= parameter is specified, then those calculations do take place. In self-summarizing mode, this parameter is not used.

If the TOPNVAR= parameter is specified, then

- input to the calculations that are defined by the TOPN set of parameters is the data set that results from the DATA step that is defined by the STACK set of parameters (if STACKVAR=YES) or the data set that results from the selection and regular subsetting DATA step (if STACKVAR= is set to or defaults to NO).
- the observations are grouped as described by the TOPNCLAS= parameter.
- a group property is calculated for each group. The property is the mean, over all observations in the group, of the value of the variable that is specified in the TOPNVAR= parameter.
- the groups are sorted based on the value of each group's mean.
- groups are chosen as specified by the TOPN= parameter.
- a macro variable that is named CPTOPNS is built whose value identifies the groups that are chosen. For example, suppose that the TOPNCLAS= parameter is set to MACHINE USER, the TOPN= parameter is set to 2, and the TOPNVAR= parameter is set to CPUTM. In this example, suppose that the TOPN calculations determine that the two groups with the highest means of CPUTM are the following:

```
machine=mach3 user=u214  <- the class values
                            identifying one
                            of the groups
machine=mach9 user=u78   <- the class values
                            identifying the
                            other of the groups
```

For this example, the macro variable is created as follows:

```
%let cptopns= (machine=mach3 &
user=u214) or
              (machine=mach9 & user=u78) ;
```

Note that CPUTM is not in the subsetting criteria.

- The summarization step obtains observations
  - that resulted from the DATA step that is defined by the STACK set of parameters (if STACKVAR=YES) or
  - that resulted from the selection and subsetting DATA step (if STACKVAR is set to or defaults to NO).

The summarization step uses a WHERE clause that is based on the CPTOPNS macro variable. The TOPN set of parameters identified the groups of interest. The WHERE clause subsets the observations so that only the observations in the identified groups are used.

VAR=*variable-list*

lists the name or names of variables to use as analysis variables in the summarization step.

If you specify STACKVAR=NO (or if it defaults to this value), then the variables must be ones that are in every view that is specified by the selection and subsetting criteria. If the STACKVAR= parameter is set to YES, then the variables

must be in one of the following lists: DURATION, the variables that are specified by the CLASS= parameter, the variable that is specified by the STACKVAL= parameter, and the variable that is specified by the STACKTYP= parameter.

In auto-summarizing mode, this parameter is required. In self-summarizing mode, this parameter is not used.

If SUMPROC=SUMMARY, then a VAR statement is generated for the PROC SUMMARY step. The statement lists the values from the VAR= parameter. Similarly, if SUMPROC=MDDB, then a VAR statement is generated for the PROC MDDB step, and the statement lists the values from the VAR= parameter.

For example, if VAR= var1 var2, then this statement is generated:

```
VAR var1 var2 ;
```

WHERE=*where-expression*
specifies the expression to use to subset the data before it is summarized. Only observations for which the expression is true will be part of the summarized data. For example, you might use a value such as WHERE= SHIFT='1' to restrict the analysis to only first-shift data.

In auto-summarizing mode, this parameter is optional. In self-summarizing mode, this parameter is required.

The WHERE expression that you specify is applied in the DATA step that is based on the selection and subsetting criteria. It is applied in a WHERE clause that is constructed as follows: the clause begins with a WHERE expression that is built from the N-*level* set of parameters, next has an AND, and then ends with the WHERE expression that is specified by the WHERE= parameter.

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. △

%CPEISSUM Notes

For auto-summarizing mode, follow these steps.

1. Within the %CPEISSUM macro, specify selection criteria by using the *tablist* parameter and at least one of the N_DETAIL=, N_DAYS=, N_WEEKS=, N_MONTHS=, and N_YEARS= parameters (the N_level set of parameters). If you use one of the N_*level* set of parameters, then in this context the use indicates that you want to prepare data from that level. (N_DETAIL= refers to the detail level, N_DAYS= refers to the day level, N_WEEKS= refers to the week level, and so on.) Do not specify values for these level parameters yet. For each combination of level and table, %CPEISSUM generates and submits a DATA step that obtains data from the view that is indicated by level_name.table_name.
2. Specify subsetting criteria in the %CPEISSUM macro by specifying nonzero values for the N_*level* set of parameters that you used for selection criteria, and by using the WHERE= parameter. Each of the N_*level* set of parameters applies to the view that contains its name. That is, the N_DETAIL= parameter applies to views that are named detail.table_name, the N_DAYS= parameter applies to views that are named day.table_name, and so on. The N_*level* set of parameters defines how much of the most recent days, weeks, months, or years of data you want from the corresponding views. The WHERE= parameter applies to each view. The WHERE= parameter specifies an expression that subsets the data that is specified by the N_*level* set of parameter(s). For each combination of level and table, the DATA step that %CPEISSUM generates and submits contains a compound WHERE clause that obtains a subset of the data in the view, based on the values of the N_*level* set of parameter(s) and the value of the WHERE= parameter.
3. Specify the transformation criteria, if any, by using the TopN and/or stack plot parameters on the %CPEISSUM macro. For TopN information, use the

TOPNVAR=, TOPN=, and TOPNCLAS= parameters. For stack plot information, use the STACKVAR=, STACKVAL=, and STACKTYP= parameters.

4 Specify the summarization criteria by using the VAR= parameter and the other summarization parameters on the %CPEISSUM macro. If you specify (or default to) SUMPROC=MDDB, then use the VAR=, CLASS=, HIER=, and STATS= parameters. If you specify SUMPROC=SUMMARY, then use the VAR=, CLASS=, and STATS= parameters.

5 Specify the output criteria as part of the summarization step or after the summarization step, by using these parameters on %CPEISSUM: OUTLIB=, OUTMEMS=, and OUTLABS=. The value of the OUTLIB= parameter in auto-summarizing mode is typically DOWNLOAD. There should be one value of the OUTMEMS= and OUTLABS= parameters for each value of the tablist parameter.

6 At this point, you have created %CPEISSUM macro code that looks similar to the following, although your code has actual values for the parameters and may not have some of the parameters that are optional for auto-summarizing mode:

```
%CPEISSUM (tablist
         , N_DETAIL= ndet
         , N_DAYS= nday
         , N_WEEKS= nwk
         , N_MONTHS= nmon
         , N_YEARS= nyr
         , WHERE= where-expression
         , TOPN= tnum
         , TOPNCLAS= tcvar
         , TOPNVAR= tvvar
         , STACKTYP= spvar
         , STACKVAL= slvar
         , STACKVAR= YES
         , VAR= varslist
         , SUMPROC= proc-name
         , CLASS= claslist <- or ,CLASS=claslist, HIERS=hierlist
         , STATS= statlist
         , OUTLIB= DOWNLOAD
         , OUTMEMS= outmlist
         , OUTLABS= outllist) ;
```

Do not submit the macro at this point. When you do submit the macro, for each view that was requested in the macro, this mode carries out the following operations one time:

- the selection and regular subsetting criteria apply to the view and result in a DATA step. The DATA step generates a data set that has all the variables that are in the view.
- if STACKVAR=YES, then the STACK set of parameters apply to the data set from the first operation and result in a DATA step. The DATA step creates a data set that has only these variables: DURATION, the variables that are specified in the CLASS= parameter, the variable that is specified in the STACKVAL= parameter, and the variable that is specified in the STACKTYP= parameter.
- if the TOPNVAR= parameter is specified, then the TOPN set of parameters apply to the data set that results from the second operation if STACKVAR=YES or to the data set that results from the first operation if STACKVAR= is set to (or defaults to) NO. The result of calculations from running the TOPN set of parameters is a set of macro variables that identify observations in the chosen groups.

- the summarization step obtains observations from the second data set (if STACKVAR=YES) or from the first data set (if STACKVAR= is set to or defaults to NO), applies the macro variables from the third operation (if TOPNVAR= is specified) to subset the observations to those in the chosen groups, and then applies the summary criteria.
- the observations are output by the summarization step according to the output criteria (typically to the library whose libref is DOWNLOAD).

To carry out the operations, this mode creates and executes SAS code that contains fragments such as these:

```
DATA DOWNLOAD.outmems# (label=outlabs) ;
  SET level_name.table_name
     (WHERE= date-range is as specified by the
             corresponding N_level parameter
             AND where-expression) ;

DATA step related to STACK set of parameters, if STACKVAR= is specified.

PROC step related to TOPN set of parameters, if TOPNVAR= is specified.

PROC step related to the VAR=, STATS=, SUMPROC=,
CLASS= (or CLASS= and HIERS=)
parameters; the procedure writes results to the locations
specified by the OUTLIB=, OUTMEMS=, and OUTLABS= parameters.
```

In the previous example, # is blank if level is detail, D if level is day, W if level is week, M if level is month, and Y if level is year, and the values of OUTLABS= and OUTMEMS= have the same positions in their lists that the table's name has in its list.

For example, suppose that the value of the tablist parameter is XTY70 XTY71. Suppose that the N_*level* parameters are N_DETAIL=5, N_DAYS=10, and N_MONTHS=2. Suppose that the value of the WHERE= parameter is SHIFT='1'. Suppose that OUTLIB=DOWNLOAD and OUTMEMS=t70 t71. Suppose that the TOPN set of parameters and STACK set of parameters are specified. Suppose that the SUMPROC=, CLASS= (or CLASS= and HIERS=), VARS=, and STATS= parameters are specified.

Then the macro will generate and execute code six times (one time for each of these views: detail.xty70, detail.xty71, day.xty70, day.xty71, month.xty70, and month.xty71). When it executes for the day.xty71 view, it will generate and execute code that looks similar to this:

```
DATA download.xty71d;
   SET day.xty71
   (WHERE= latest-10<datepart(datetime)<latest  AND shift='1');

DATA step related to STACK set of parameters,
if STACKVAR=YES is specified.

PROC step related to TOPN set of parameters,
if TOPNVAR= is specified.

PROC step that summarizes the data as specified by the VAR=, STATS=,
SUMPROC=, CLASS= and HIERS= parameters, and directs the output to
the library specified by the DOWNLOAD libref and into its module
whose name is xty71d.
```

The overall result will be six modules. In this context, module means MDDB or data set. If SUMPROC=MDDB, then the result will be six MDDBs (one for each view). If SUMPROC=SUMMARY, then the result will be six data sets (one for each view).

7 Define a libref named DOWNLOAD as a pointer to the library to which the results are output. You can use the SAS LIBNAME statement to create the libref. Here's an example:

```
LIBNAME DOWNLOAD 'location-of-the-output-library' DISP=OLD ;
          (on z/OS)
LIBNAME DOWNLOAD 'location-of-the-output-library' ;
          (on UNIX or Windows)
```

The output library must already exist. This is the library from which you will download the data.

8 Create a SAS job that executes %CPSTART, creates the DOWNLOAD libref, and executes %CPEISSUM (using the macro that you created earlier in this process).

For batch job examples of using %CPEISSUM in auto-summarizing mode, see the "Examples" section of this macro. Also, as described at the beginning of the "Notes" section of the %CPEISSUM macro, see the CMEISDWN member in CPMISC.

9 Submit the job that you have created and check the results.

10 Schedule the job to run daily (after the daily process and reduce job). Or move all but the %CPSTART code to the end of the daily process-and-reduce job.

For self-summarizing mode, follow these instructions:

1 Create %CPEISSUM macro code that specifies selection criteria by using the tablist parameter and at least one of the N_DETAIL=, N_DAYS=, N_WEEKS=, N_MONTHS=, and N_YEARS= parameters.

If you use an N_*level* parameter, then in this context the use indicates that you want to prepare data from that level. (N_DETAIL= refers to the detail level, N_DAYS= refers to the day level, N_WEEKS= refers to the week level, and so on.) Do not specify values for the N_*level* parameters yet.

For each combination of level and table, %CPEISSUM generates and submits a DATA step that obtains data from the view that is indicated by level_name.table_name. You can specify additional selection criteria in the SAS catalog entries.

2 In this %CPEISSUM macro, specify subsetting criteria by specifying nonzero values for the N_*level* set of parameters that you used for selection criteria and by using the WHERE= parameter.

The N_*level* set of parameters apply to the views that contain their names. That is, the N_DETAIL= parameter applies to views that are named detail.table_name, the N_DAYS= parameter applies to views that are named day.table_name, and so on. The N_*level* set of parameters define how much of the most recent days, weeks, months, or years of data you want from the corresponding views.

The WHERE= parameter applies to each view. The WHERE= parameter specifies an expression that subsets the data that is specified by the N_*level* parameter(s).

For each combination of level and table, the DATA step that %CPEISSUM generates and submits contains a compound WHERE clause that obtains a subset of the data in the view, based on the values of the N_*level* parameter(s) and the value of the WHERE= parameter.

3 In this %CPEISSUM code, do not specify the transformation criteria. You can specify transformation criteria, if any, in the SAS catalog entries that contain the prewritten code that is to be submitted.

4 In this %CPEISSUM code, you do not specify the summarization criteria.

(You will specify the summarization criteria in the SAS catalog entries that contain the prewritten code that is to be submitted.)

5 In this %CPEISSUM macro, use the SOURCAT= parameter to specify the location of the SAS catalog that contains one or more entries with prewritten code that summarize the data (and do whatever else is required).

6 In this %CPEISSUM macro, as part of the output criteria, use the OUTLIB=, OUTMEMS=, and OUTLABS= parameters to specify the output locations to which the selection and subsetting code (as specified by the parameters in steps 1 and 2) is to write its results, which are intermediate results.

The value of the OUTLIB= parameter in self-summarizing mode is typically WORK. There should be one value of the OUTMEMS= and OUTLABS= parameters for each value of the tablist parameter.

7 At this point, you have created a call to %CPEISSUM macro code that looks similar to the following, although your call has actual values for the parameters and might not have some of the parameters that are optional for self-summarizing mode:

```
%CPEISSUM (tablist,
          ,N_DETAIL= ndet
          ,N_DAYS= nday
          ,N_WEEKS= nwks
          ,N_MONTHS= nmts
          ,WHERE= where-expression
          ,SOURCAT= sref.scat
          ,OUTLIB= oref
          ,OUTMEMS= outmems-list
          ,OUTLABS= outlabs-list) ;
```

Do not submit the macro yet. But when you do submit the macro, for each view that was requested in the %CPEISSUM code, this mode carries out the following operations one time:

- first, the selection and regular subsetting criteria apply to the view and result in a DATA step. The DATA step generates a data set that has all the variables that are in the view and is named by using the OUT set of parameters.
- second, the appropriate entry in the source catalog is submitted. The entry is written to execute the summarization step and any other code in the entry and result in a data set or MDDB in the library whose libref is DOWNLOAD.

To carry out the operations, this mode creates and executes SAS code that contains fragments such as these:

```
DATA  &outlib.&outmems# (label="&outlabs") ;
   SET &level.&tablenm (where &whdate &whparm)) ;

Submits the code in sref.scat.outmems#.source
```

In this example, # is blank if level is detail, D if level is day, W if level is week, M if level is month, and Y if level is year; &whdate is the value of the WHERE expression that is based on the values of the N_*level* set of parameters; &whparm is the value of the WHERE expression that is based on the WHERE= parameter and prefixed with an AND; and the position of OUTMEMS and OUTLABS in their lists corresponds to the position of *tablenm* in the *tablist*.

For example, suppose that the value of the *tablist* parameter is XTY70 XTY71. Suppose that the N_*level* set of parameters are N_DETAIL=5, N_DAYS=10, and N_MONTHS=2. Suppose that the value of the WHERE= parameter is SHIFT='1'. Suppose that OUTLIB=WORK and OUTMEMS=t70 t71. Suppose that the SOURCAT= parameter is WORK.TEMP.

Then the macro will generate and execute code six times (one time for each of these views: **detail.xty70**, **detail.xty71**, **day.xty70**, **day.xty71**, **month.xty70**, and **month.xty71**). When it executes for the **day.xty71** view, it will generate and execute code that looks similar to this:

```
DATA work.xty71d ;
   SET day.xty71 (WHERE= latest-10<datepart(datetime)<latest AND shift='1');

Submits the code in work.temp.xty71d.source
```

8 For one level of one table from which you are requesting data, write code that creates a SAS catalog entry that is called sref.scat.outmems#.source, where the values **sref**, **scat**, **outmems**, and **#** have the same meanings as in the previous step. Here's an example:

```
%CPCAT; CARDS4;
;;;;
%CPCAT ( CAT=sref.scat.outmems#.source);
```

Do not submit the code yet.

9 In that code, insert the prewritten code (for views that are based on the # level of the tablenm table) that summarizes the data that was selected and subset in the DATA step that %CPEISSUM generated, and does whatever else is required. Here's an example:

```
%CPCAT; CARDS4;
prewritten code for summarizing data from oref.outmems#; the code's
last statement is typically RUN; (or RUN; QUIT; if the summarizing
procedure requires a QUIT;)
;;;;
%CPCAT ( CAT=sref.scat.outmems#.source);
```

For example, the SAS catalog entry prewritten code in WORK.TEMP.XTY70D.SOURCE (that is, for the day level of the XTY70 table) might look like this:

```
proc summary data=work.xty70d ... ;
    ... ;
    output out=download.xty70d;
    ... ;
    run;
    quit;
```

It might also look like this:

```
proc mddb data=work.xty70d out=download.xty70d ... ;
... ;
run; quit;
```

Additional code can be in front of or behind the summarization code. In general, the SAS catalog entry can contain calls to SAS IT Resource Management macros and can contain SAS statements. However, do not do anything in the prewritten code for this level of this table that prevents the prewritten code from running correctly for other levels and other tables.

Reminder: In the SAS catalog entry, whatever code obtains the data must obtain it from the &outlib.&outmems# location that is specified in the DATA step that %CPEISSUM generates, and whatever code writes the results must write them to the location that is specified by the DOWNLOAD libref into the modules (that is, data sets or MDDBs) whose names are specified by &outmems#.

Do not submit the code yet.

10 Repeat steps 8 and 9 for each of the remaining level-and-table combinations from which you are requesting data.

11 Write the code that creates a libref for the library to which you want to output the intermediate results (the results that the %CPEISSUM parameters themselves create), if it does not already exist. (The WORK libref already has been defined by SAS.)

Also write the code that creates a libref for the library to which you want to output the final results (the results that the code in the SAS catalog entries create), if it does not already exist. The libref of the final library should be DOWNLOAD.

If the prewritten code uses additional libraries, then also write the code that creates the librefs for those libraries. (The call to %CPSTART creates a number of librefs. If you want to use one of those librefs, then you do not need to create it.) You can use the SAS LIBNAME statement. For example, to define DOWNLOAD as the libref of the final library, use the following:

```
LIBNAME DOWNLOAD '  location-for-final-library' DISP=OLD ;  (on z/OS)
LIBNAME DOWNLOAD '  location-for-final-library' ; (on UNIX or Windows)
```

The final library must already exist. This is the library from which you will download the data.

12 Create a SAS job that calls %CPSTART, creates one or more librefs if they do not already exist (by using your code from step 11), creates the SAS catalog entries (by using your code from steps 8 through 10), and then calls %CPEISSUM (by using your code from steps 1 through 6 and reviewed in step 7).

For examples of batch jobs that contain self-summarizing-mode calls to %CPEISSUM, see the "Examples" section of this macro. Also, as described at the beginning of %CPEISSUM "Notes" section, see the CMEISDWN member in CPMISC.

13 Submit the job and check the results.

14 If the job runs correctly, then create and submit a job (with calls to %CPSTART, a LIBNAME statement if the libref does not already exist, and calls to %CPCAT) that writes the prewritten code to a more permanent location, so that loading the SAS catalog entries does not need to take place every day. Then, from the job that you created in steps 1 through 13, remove the loading of the temporary SAS catalog entries and change the values of the SOURCAT= parameters to point to the new permanent locations.

Do not store prewritten code in PGMLIB, because PGMLIB is replaced when you install SAS IT Resource Management maintenance.

Remember: When you store the prewritten code, you must have write access to the SAS library where you are storing it. To request write access, use DISP=OLD or ACCESS=WRITE on the LIBNAME statement that defines the libref for the library.

15 Schedule the job to run daily (after the daily process-and-reduce job). Or move all but the %CPSTART call to the end of the daily process-and-reduce job.

%CPEISSUM Examples

Example 1: Auto-Summarizing Mode

```
* From an HP OpenView PDB,
* select and summarize the most recent 7 days of
* workstation general data from the day level of table PCSGLB
* by using PROC SUMMARY under UNIX
* resulting in a SAS data set named glbwD
* (with the label Global WorkStation Data)
* in directory /tmp/dtr/cpu/download. ;

* Start SAS IT Resource Management and specify the active PDB ;
    %CPSTART( pdb=/tmp/pdb-workstations );

  * Create a libref for the final output library ;
    libname download "/tmp/dtr/cpu/download";

  * Select the data and summarize it ;
    %CPEISSUM( pcsglb,
               n_days= 7,
               var = glbcpto glbcsht glbdkto glblnco
                     glbnrq glbpgrt glbuser ,
               sumproc= summary,
               class= machine date hour datetime shift,
               stats= mean min max,
               outlib= download,
               outmems= glbw,
               outlabs= 'Global Workstation Data' );
```

Example 2: Auto-Summarizing Mode

```
* From an HP OpenView PDB,
* select and summarize the most recent 7 days of
* server general data from the day level of table PCSGLB
* by using PROC MDDB under UNIX
* resulting in a MDDB named glbsD
* with the label Global Server Data
* in directory /tmp/dtr/cpu/download. ;

* Start SAS IT Resource Management and specify the active PDB;
     %cpstart ( pdb=/tmp/pdb-servers );

* Create a libref for the final output library;
     libname download "/tmp/dtr/cpu/download";

* Select data and summarize it;
    %cpeissum ( pcsglb,
                n_days= 7,
                var = glb1rep glbapro glbcpto glbcsht glbdkto glbfmem
                      glbidle glblnco glbnrq glbpgrt glbphrd glbphwt
                      glbpron glbprtm glbsmem glbsys glbtran glbtrnm
                      glbuser glbusrm glbusrn glbvirm ,
                 sumproc= mddb,
```

```
                class=machine date hour datetime shift,
                hiers=Machine Date Hour ! Machine Datetime,
                stats= mean min max,
                outlib=download,
                outmems=glbs,
                outlabs='Global Server Data' ) ;
```

Example 3: Auto-Summarizing Mode

```
* From an HP OpenView PDB,
* select and summarize the most recent 7 days of
* server cpu data from the day level of table PCSGLB
* by using PROC MDDB under UNIX
* resulting in a MDDB named glbscpuD
* with the label Global Server CPU Data
* in directory /tmp/dtr/cpu/download
* and also prepare for stack plots. ;

* Start SAS IT Resource Management and specify the active PDB;
    %CPSTART( pdb=/tmp/pdb-servers ) ;

* Create a libref for the final output library;
    libname download "/tmp/dtr/cpu/download";

* Select data, prepare for stackplots, and summarize the data;
    %CPEISSUM( pcsglb,
               n_days= 7,
               stackvar=yes,
               stackval=sval,
               stacktyp=styp,
               var = glbcsw glbint glbscut glbrtim glbnice glbnmut,
               sumproc= mddb,
               class= machine date hour datetime shift,
               hiers= Machine Date Hour ! Machine Datetime,
               stats= mean min max,
               outlib= download,
               outmems= glbscpu,
               outlabs= 'Global Server CPU Data') ;
```

Example 4: Self-Summarizing Mode

```
*  From an HP OpenView PDB,
*  select and summarize the most recent 7 days of
*  workstation general data from the day level of table PCSGLB
*  by using PROC SUMMARY under UNIX,
*  resulting in a SAS data set named pcsglbd
*  in directory /tmp/dtr/cpu/download. ;

* Start SAS IT Resource Management and specify the active PDB:
    %CPSTART( pdb=/tmp/pdb-workstations );

* Create a libref for the final output library ;
    libname download "/tmp/dtr/cpu/download" ;

* Create a catalog entry that will obtain data created by the DATA step
```

```
* that %CPEISSUM generates, and summarize the data by using PROC SUMMARY;
    %cpcat;
       cards4;
       proc summary data=work.pcsglbd;
       class machine date hour datetime shift;
       var glbcpto glbcsht glbdkto glblnco
           glbnrq glbpgrt glbuser ;
       output out=download.pcsglbd
              mean= min= max= ;
       run ;
       quit ;
   ;;;;
   %cpcat ( cat= work.temp.pcsglbd.source) ;

* Select the data and submit the catalog entry ;
   %CPEISSUM( pcsglb,
              n_days= 7,
              sourcat= work.temp,
              outlib= work );
```

%CPFAILIF

Based on the value of a specified expression, terminates the SAS IT Resource Management session and the SAS session in which SAS IT Resource Management is running

%CPFAILIF Overview

The %CPFAILIF macro evaluates an expression. If the expression is evaluated as true, then %CPFAILIF terminates the SAS IT Resource Management session and the SAS session in which SAS IT Resource Management is running. To execute additional calls to SAS IT Resource Management, you must start a new SAS IT Resource Management session (and a new SAS session).

%CPFAILIF Syntax

%CPFAILIF(
expression
<,**CODE=***exit-code*>);

Details

expression
specifies the expression that you want %CPFAILIF to evaluate. The expression must be a valid SAS expression and must use symbols such as GT, GE, LT, LE, EQ, or NE instead of the special symbols >, >=, <, <=,=, or ^=. Specify the expression so that it will evaluate as true if the task you are evaluating did not execute properly, and false if the task did execute properly. For example, if you are evaluating whether the %CPSTART macro executed properly, then you would want to specify the expression such that if the expression is true, it will indicate that the %CPSTART macro did not execute properly.

This positional parameter is required and must be specified in the first position.

If the expression evaluates as true, then %CPFAILIF terminates both the SAS IT Resource Management session and the SAS session and provides an exit code (if specified in the CODE= parameter) to the operating system. If the expression evaluates as false (the previous task executed properly), then %CPFAILIF passes control to the next statement.

CODE=*exit-code*
an integer that will be reported to the operating system as the exit code if the expression specified in the *expression* parameter evaluates as true. This parameter is not required. *The default value is 16.*

%CPFAILIF Notes

CAUTION:
This macro is designed for use in true batch mode. △

If you select **Locals ▶ Submit** from the SAS Program Editor window to submit a SAS program with this macro, and the value of the expression causes the macro to terminate both the SAS IT Resource Management session and the associated SAS session, then the SAS window interface will also close.

To prevent this from happening, use the %CPDEACT macro in place of the %CPFAILIF macro when running within the SAS Program Editor window. For more information, see "%CPDEACT" on page 84.

%CPFAILIF Example

In this example, the first call to %CPSTART starts a SAS IT Resource Management session, activates a PDB, and performs other tasks. The first call to %CPFAILIF verifies that %CPSTART completed successfully. (A return code of zero represents successful completion; a nonzero return code represents a problem.)

If the value of the return code is not zero (that is, if %CPSTART had a problem), then %CPFAILIF terminates the SAS IT Resource Management session and the SAS session. If the value of the return code is zero (that is, if %CPSTART completed successfully), then %CPDEACT does not terminate the SAS session.

If the SAS IT Resource Management session and the SAS session are terminated, then %CPPROCES does not execute because neither session is running. If neither session is terminated, then %CPPROCES executes. The remaining pairs of %CPFAILIF and other macro combinations operate the same way. Because %CPRUNRPT has no _RC= parameter, the next statement executes regardless of the status of the %CPRUNRPT macro when it completes.

The second call to %CPSTART cannot cause the SAS IT Resource Management session or SAS session to recover if an earlier call caused them to terminate, because in that case the second call to %CPSTART is never reached.

```
...
%CPSTART ( ... , _rc=myrc) ;
%CPFAILIF ( &myrc ne 0, code=999) ;
* terminates SAS if %CPSTART was not successful ;
%CPPROCES (..., _rc=myrc )  ;
%CPFAILIF ( &myrc ne 0, code=999) ;
* terminates SAS if %CPPROCES was not successful ;
%CPREDUCE (..., _rc=myrc) ;
%CPFAILIF ( &myrc ne 0, code=999) ;
* terminates SAS if %CPREDUCE was not successful ;
```

```
%CPRUNRPT (...) ;
   ...
%CPSTART ( ..., _rc=myrc) ;
%CPFAILIF ( &myrc ne 0, code=999) ;
* terminates SAS if %CPSTART was not successful ;
 ...
```

%CPHDAY

Updates the active PDB's shift code for holidays and list of holidays

%CPHDAY Overview

The %CPHDAY macro enables you to add, delete, and modify the list of site holidays. You can also use this macro to change the shift code for active holidays, by using the HSHIFT= parameter.

%CPHDAY Syntax

%CPHDAY(
<**ACTIVE=**(*date,date...*)>
<,**ADD=**(*date="label",date="label"...*)>
<,**DELETE=**(*date,date,...*)>
<,**HSHIFT=***shift*>
<,**INACTIVE=**(*date,date...*)>);

Details

ACTIVE=*(date,date...)*
specifies the list of holidays that you want to make active for the active PDB. Enclose the list of dates in parentheses and use commas to separate the dates in the list.

By default, when you add a new holiday, it is active unless you specify the INACTIVE= parameter with that holiday.

A holiday may be specified in both the ADD= parameter and either ACTIVE= or INACTIVE= parameters, in which case the holiday will be added and then the activity status will be modified accordingly. Specifying the same holiday as both active and inactive causes unpredictable results.

See the "Notes" section of this macro for information on the date format.

ADD=*(date="label",date="label"...)*
lists the dates and labels of the holidays that you want to add to the active PDB holiday list. Enclose the paired list of dates and labels in parentheses and use commas between the pairs of dates/labels in the list. By default, when you add a new holiday it is active unless you specify the INACTIVE= parameter with that holiday.

A holiday may be specified in both the ADD= parameter and either ACTIVE= or INACTIVE= parameters, in which case the holiday will be added and then the activity status will be modified accordingly. Specifying the same holiday as both active and inactive causes unpredictable results.

See the "Notes" section of this macro for information on the date format.

Do not use the equal sign (=) or commas (,) within the label text.

DELETE=*(date,date...)*
lists the dates of the holidays that you want to delete from the active PDB's holiday list. Enclose the list of dates in parentheses and use commas to separate the dates in the list. See the "Notes" section of this macro for information on the date format.

HSHIFT=*shift*
specifies a single-byte character that will be used to represent all shifts on all active dates in the holiday list. By indicating which days are holidays (when activity may be low), you can choose to include or exclude these days when you generate reports on your data.

shift can be any character except the ampersand (&), the percent sign (%), or quotes, either single or double (' or ").

The holiday shift is a special instance of the shift definition. If you set the holiday shift to '2', for example, then you are saying that any time an incoming transaction falls on an active holiday, the value of the SHIFT variable is set to 2. The same shift can be used as both a holiday shift and for other purposes (although the default definitions do use the same shift code for multiple purposes). If you set the holiday shift to 2, and also set all day Saturday and Sunday to 2, then any observation that falls on either a holiday or a weekend will get SHIFT=2. Transaction reports will reflect this shift-based grouping.

The default value of HSHIFT= is zero. If you specify a different value for the shift code for holidays, then remember to change the shift descriptions ("labels") using the SHIFTDSC= parameter on the %CPPDBOPT macro.

INACTIVE=*(date,date...)*
specifies the list of dates that you want to deactivate. Deactivated dates are retained in the holiday list, but they are assigned shift codes from the workshift grid and not the holiday shift code.

Enclose the list of dates in parentheses and use commas to separate the dates in the list.

A holiday may be specified in both the ADD= parameter and either ACTIVE= or INACTIVE= parameters, in which case the holiday will be added and then the activity status will be modified accordingly. Specifying the same holiday as both active and inactive causes unpredictable results.

See the "Notes" section of this macro for information on the date format.

%CPHDAY Notes

All dates that are specified within this macro should use a legal, SAS date-constant format. The format for valid SAS dates requires that the date be enclosed in quotation marks. For example, **'01-JAN-01'd** is an example of a valid SAS date for January 1, 2001, and **'01-JAN-2002'd** is an example of a valid SAS date for January 1, 2002. Do not use datetime-constants such as **'01-JAN-03:23:30'dt.** If the date format is valid but the date is unknown, then the date is ignored. If the date format is invalid, then the macro will fail.

SAS supports both two-digit and four-digit years. Therefore, year 2001 can be represented in SAS as either 01 or 2001, if the SAS system option YEARCUTOFF is set appropriately.

If your site does not specify a holiday list and holiday shift code, then SAS IT Resource Management uses the default holiday list and holiday shift code from SITELIB.

When you run this macro, the updates take place immediately. This means that macros that you run later in the job (after this macro) will use the updated values that are set by this macro, not the old values.

%CPHDAY Example

The following example deletes the holiday December 25, 2000. The date December 25, 2001, is added as a holiday and labeled Christmas. The date January 1, 2002, is added as a holiday and labeled New Years Day. The already existing date December 31, 2001, is activated as a holiday. All other existing holidays for the active PDB are unmodified.

```
%CPHDAY(delete=("25-DEC-00"d),
        add=("25-DEC-2001"d="Christmas",
             "01-JAN-2001"d="New Years Day"),
        active=("31-DEC-2001"d)  );
```

%CPINSPKG

Installs the contents of a SAS IT Resource Management package into an existing SAS IT Resource Management installation

%CPINSPKG Overview

The %CPINSPKG macro reads the contents of a SAS IT Resource Management package. For example, if the %CPPKGCOL macro created the package, the package contains user-written collector-support entities for a particular data collector or data source.

Then, the %CPINSPKG macro writes the contents of the package to the appropriate places in an existing SAS IT Resource Management installation. For example, if the %CPPKGCOL macro created the package, then after the installation the user-written collector-support entities reside in the specified destination and are in the same search paths as the SAS IT Resource Management supplied collector-support entities.

%CPINSPKG Syntax

%CPINSPKG(
DEST=ADMIN | SITELIB | PGMLIB
,**DOCLOC=***location-of-report*
,**PACKFILE=***fileref*
,**REPLACE=**REPLACE | NOREPLACE | APPEND | MERGE
<,**FORCE=**NO | SOFTWARE | DICTIONARY | ALL>
<,**_RC=***macro-var-name*>);

Details

DEST=ADMIN | SITELIB | PGMLIB
specifies the libref of the library where the contents of the package are to be written.

This parameter is required.

The valid values are

- ADMIN, which is the libref of the ADMIN library in the active PDB. Installing to ADMIN means that the contents of the package, after installation, are available to users who have read or read/write access to this PDB. Note that you must have write access to ADMIN to install into ADMIN.

- SITELIB, which is the libref of the active site library. Installing to SITELIB means that the contents of the package, after installation, are available to users who have read or read/write access to this site library. Note that you must have write access to SITELIB to install into SITELIB.
- PGMLIB, which is the libref of the SAS IT Resource Management program library. Installing to PGMLIB means that the contents of the package, after installation, are available to users who have read or read/write access to the program library (and, depending on the operating system, related libraries in which SAS IT Resource Management supplied software resides). Note that you must have write access to PGMLIB to install into PGMLIB.

DOCLOC=*location-of-report*
specifies the location where the report is to be written.

Note: Do not use quotation marks around the location. △
This parameter is required.
The form for specifying the location is operating system dependent:

- On UNIX and Windows, specify the full path and name of a directory. The directory must already exist.
- On z/OS, specify the fully qualified name of a PDS or specify the full path and name of a directory in the z/OS UNIX File System area. The PDS or directory must already exist.

PACKFILE=*fileref*
specifies a fileref. The fileref points to the external file where the package will be stored (if the package is being created) or is stored (if the package is being reported on or installed). The fileref must already exist by the time that the macro is called.

For example, suppose that you are creating in Windows a package of collector-supported entities and you want to use a fileref of **pkgfile**. Before your call to the %CPPKGCOL macro, you must have a SAS FILENAME statement similar to this one:

```
FILENAME pkgfile 'c:\ucollpkg.xpt' ;
```

where **c:\ucollpkg.xpt** is the location where the package is to be written.

Similarly, suppose that you are reporting on the same package, which has been FTP'd in binary mode to UNIX, and you want to use a fileref of **unixpack**. Before your call to the %CPRPTPKG macro, you must have a SAS FILENAME statement similar to this one:

```
FILENAME unixpack '\usr\itsv\ucoll\ucollpkg.xpt' ;
```

where **\usr\itsv\ucoll\ucollpkg.xpt** is the location of the package.

REPLACE=REPLACE | NOREPLACE | APPEND | MERGE
specifies how the package is to be installed.
This parameter is required. The default value is NOREPLACE.
The valid values are

- REPLACE, which means that first the macro looks for the "subject" of the package. Next, all of the destination's entities that correspond to that "subject" are removed. Finally, all the entities in the package are installed in the destination library.

  For example, if the package was created by %CPPKGCOL, then the %CPINSPKG macro first determines which collector name and tool name are represented in the package. Next, all collector-support entities in the destination that are "tagged" with that collector name and tool name combination are deleted. Then the collector-support entities in the package are installed in the destination library.

- NOREPLACE, which means that first the macro looks for the "subject" of the package. Next, the destination is searched for any entities that correspond to that "subject." If any are found, the installation stops. If none are found, the entities in the package are installed in the destination library.

  For example, if the package was created by %CPPKGCOL, then the %CPINSPKG macro first determines which collector name and tool name are represented in the package. Next, the %CPINSPKG macro searches for any collector-support entities in the destination that are "tagged" with that collector name and tool name combination. If any such entities are encountered in the destination, installation stops. If none are encountered in the destination, then the collector-support entities in the package are installed in the destination library.
- APPEND, which means that
  - if any entity in the package corresponds to an entity already in the destination, the entity in the destination is not replaced and the entity in the package is ignored
  - if any entity in the package has no corresponding entity already in the destination, the entity in the package is installed
  - if there is an entity in the destination that does not have a corresponding entity in the package, the entity in the destination is not removed.

  For example, suppose that you used %CPPKGCOL to create a package for a certain data collector, and the package contained one table definition, staging code, and documentation. Suppose that you installed it. Now suppose that you use %CPPKGCOL to create a revised package for that collector, and the package contains two table definitions (a second table and a revised version of the first table), revised staging code, and a format. If you use REPLACE=APPEND for the second installation, after the second installation the destination library will contain the old version of the first table definition, the (new) definition of the second table, the old version of the staging code, the (new) format, and the (old) documentation.
- MERGE, which means that
  - if any entity in the package corresponds to an entity already in the destination, the entity in the destination is removed and the entity in the package is installed
  - if any entity in the package does not correspond to an entity already in the destination, the entity in the package is installed
  - if there is an entity in the destination that does not have a corresponding entity in the package, the entity in the destination is not removed.

  For example, suppose that you used %CPPKGCOL to create a package for a certain data collector, and the package contained one table definition, staging code, and documentation. Suppose that you installed it. Now suppose that you use %CPPKGCOL to create a revised package for a certain collector, and the package contains two table definitions (a second table and a revised version of the first table), revised staging code, and a format. If you use REPLACE=MERGE for the second installation, after the second installation the destination library will contain the new version of the first table definition, the (new) definition of the second table, the new version of the staging code, the (new) format, and the (old) documentation.

For more information about the behavior of this parameter, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

FORCE= NO | SOFTWARE | DICTIONARY | ALL
specifies the action that is to occur if the SAS IT Resource Management software version and/or dictionary version under which the package was created does not match that of the destination.

This parameter is optional. The default value is NO.

The valid values are

- NO, which means that the installation is to stop if the software versions do not match and/or if the dictionary versions do not match
- SOFTWARE, which means that the installation is to continue even if the software versions do not match, as long as the dictionary versions do match
- DICTIONARY, which means that the installation is to continue if the dictionary versions do not match, as long as the software versions do match
- ALL, which indicates that the installation is to continue even if the software versions do not match and the dictionary versions do not match.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPINSPKG Notes

If the package needs to be transported from the system where it was created to another system where it is to be installed, use a mechanism such as binary mode in FTP, that does not do any conversion.

The search path starts with the library to which the ADMIN libref points. If the item that is being searched for is not found in ADMIN, the library to which the SITELIB libref points is searched. If the item that is being searched for is not found in SITELIB, then the library to which the PGMLIB point is searched.

For more information about collectors and collector support, see the section "Setting Up the Server" in the chapter "Setup: Introduction" in the *SAS IT Resource Management User's Guide*. For more information about packaging and installing collector support, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*, "%CPPKGCOL" on page 144, and "%CPRPTPKG" on page 177.

Note that this macro (%CPINSPKG) generates another copy of the report that the %CPRPTPKG macro generates. The %CPRPTPKG report is typically used for

temporary documentation. The %CPINSPKG report is typically used for permanent documentation. Both reports are identical and in HTML.

%CPINSPKG Examples

Example 1

The following example demonstrates installing a package to PGMLIB. The REPLACE=NOREPLACE option means that nothing is to be replaced. If the destination contains anything that corresponds to the package, the installation stops. For example, if the package was created with %CPPKGCOL and the destination contains anything for the collector name and tool name that are represented by the package, then the installation stops.

```
filename pkgfile 'c:\packfile.xpt' ;

%cpinspkg(packfile=pkgfile
        , dest=pgmlib
        , replace=noreplace
        , docloc=c:\doc
        , _rc=insrc) ;

%put CPINSPKG return code is &insrc ;
```

Example 2

The following example demonstrates installing a package to PGMLIB on z/OS. The REPLACE=REPLACE setting means that anything related to the contents of this package is deleted before the package is installed. For example, if the package was created with %CPPKGCOL and the destination contains anything for the collector name and tool name represented by the package, everything in the destination for that collector name and tool name is deleted and then the collector-support entities in the package are installed.

```
filename pkgfile 'prod.itsv.pack' ;

%cpinspkg(packfile=pkgfile
        , dest=pgmlib
        , replace=replace
        , docloc=prod.pack.doc
        , _rc=insrc) ;

%put CPINSPKG return code is &insrc ;
```

%CPLOGRC

Redirects the SAS log to a temporary file and scans the file for error messages (UNIX and Windows only)

%CPLOGRC Overview

The %CPLOGRC macro enables you to supplement the standard checking of return codes in batch mode. You can use the %CPLOGRC macro to write your SAS log to a temporary file and then scan that file for strings of text that you specify.

%CPLOGRC Syntax

%CPLOGRC(
opcode
<,**DELIM=***delimiter*>
<,**_HOWMANY=***macro-variable-name*>
<,**LOOKFOR=***list-of-strings*>
<,**_RC=***macro-var-name*>
<,**WHEREAT=***search-locations*>);

Details

opcode
indicates the action that the %CPLOGRC macro will perform. You can use more than one sequence of calls (a sequence of calls has START...CHECK(s)...STOP) to the %CPLOGRC macro.

A sequence must not directly or indirectly execute a SAS PROC PRINTTO statement. (Because all SAS IT Resource Management report macros use PROC PRINTTO, do not include calls to the report macros within a %CPLOGRC sequence.)

Each call to the %CPLOGRC macro can specify (with the _RC= parameter) the name of the macro variable to which the macro will write the return code from that call.

Regardless of whether the _RC= parameter is specified on each call to %CPLOGRC, the %CPLOGRC macro also uses a global macro variable that is named CPLRMAX in order to keep track of the highest value of the return code from %CPLOGRC during the current SAS session. The value of CPLRMAX is not reset by START. Thus, at the end of the SAS session, you can use the value of CPLRMAX as one measure of success over all the sequences that are used in the SAS session.

This parameter is required.

START
indicates that the macro will redirect to a temporary file the log stream that is controlled by the -log option. The only other %CPLOGRC parameter that can be used with the START value of the opcode parameter is the _RC= parameter. In the macro variable that is specified by the _RC= parameter, the return code from calling %CPLOGRC with the opcode parameter set to START will have one of the following values:

- zero - successful completion
- nonzero - there was a problem.

For more explanation of the return codes, see messages in the log that was specified by the SAS -altlog option.

CHECK
indicates that %CPLOGRC is to finish writing out any buffers that contain the log stream and then check the temporary file for one or more strings that

are specified by the LOOKFOR= parameter. The range of the check is from the start of the file to the current end of the file.

All other %CPLOGRC parameters can be used with the CHECK value of the opcode parameter. In the macro variable that is specified by the _RC= parameter, the return code from calling %CPLOGRC with the opcode parameter set to CHECK will have one of the following values:

- 0 - none of the specified strings were found, and the call to %CPLOGRC completed successfully
- 4 - one or more of the specified strings were found, and the call to %CPLOGRC completed successfully
- 8 - none of the specified strings were found, but some problem occurred during the call to %CPLOGRC
- 12 - one or more of the specified strings were found, but some problem occurred during the call to %CPLOGRC.

For more explanation of the return codes, see messages in the log that was specified by the SAS -altlog option

STOP
indicates that %CPLOGRC is to resume sending the log stream that is controlled by the -log option to its normal location and is to delete the temporary file that was created by the previous START. The only other %CPLOGRC parameter that can be used with the STOP value of the opcode parameter is the _RC= parameter.

In the macro variable that is specified by the _RC= parameter, the return code from calling %CPLOGRC with the opcode parameter set to STOP will have one of the following values:

- zero - successful completion
- nonzero - there was a problem.

For more explanation of the return codes, see messages in the log that was specified by the SAS -altlog option.

DELIM=*delimiter*
specifies the delimiter that separates the listed items in the LOOKFOR= parameter. This parameter is optional, and *the default is one or more blank spaces.*

This parameter is available only when the value of the opcode parameter is CHECK.

_HOWMANY=*macro-variable-name*
specifies the name of the macro variable that contains the list of counts of the number of occurrences of each string in the search location. If there is more than one count, then the counts are separated by one or more blanks. (The value of the DELIM= parameter does not affect the delimiter in the list of counts.) The list can be a maximum of 32,767 characters long.

You can use a SAS macro, such as %SCAN, to find each count. For example, if the macro variable name is *counts*, then you can find the second count in the list by using **`%SCAN (&counts, 2)`**. *By default, if you do not specify this parameter, then the count information is not retained.* There is no default macro variable name.

This parameter is available only when the value of the opcode parameter is CHECK.

The list is sensitive to position. Each count in the _HOWMANY= parameter list corresponds to the string in the same position in the LOOKFOR= parameter list and to the location in the same position in the WHEREAT= parameter list.

LOOKFOR=*list-of-strings*

lists the strings to search for in the temporary file that is created by the opcode START. The search is case sensitive. The list of characters can be a maximum of 32,767 long. *The default value is ERROR: and the default delimiter is a blank space*. You can specify a different delimiter by using the DELIM= parameter.

If there is more than one string, then separate the strings with the delimiter. If a string includes the default delimiter, then specify a different delimiter.

This parameter is available only when the value of the opcode parameter is CHECK.

_RC=*macro-var-name*

specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

WHEREAT=*search-locations*

lists the search locations for the strings that are specified in the LOOKFOR= parameter. Valid search locations (values for this parameter) are

- BEGINLINE - the string must start in the first position on the line
- ENDLINE - the string must end in the last position on the line
- ANYWHERE - the string can be anywhere on the line.

The default value is BEGINLINE.

If there is more than one location, then separate the locations with one or more blanks. The value of the DELIM= parameter does not affect the delimiter in the list of locations. The list of locations can be a maximum of 32,767 characters in length.

The list is sensitive to position. Each keyword in the WHEREAT= parameter list corresponds to the string in the same position in the LOOKFOR= parameter list and to the count in the same position in the _HOWMANY= parameter list.

This parameter is available only when the value of the opcode parameter is CHECK.

%CPLOGRC Notes

CAUTION:

This macro is designed for use only in true batch mode. Do not submit this macro through the SAS Program Editor window.

If the SAS -altlog option is not used and/or the CPLOGRC macro variable is not set to one of the legal values, then calls to the %CPLOGRC macro are ignored. You must use the -altlog option when you start SAS. Here's an example:

```
  sas -altlog /tmp/myalt.log
```

△

The -altlog option causes there to be two streams that contain the SAS log. The %CPLOGRC macro redirects the stream that is controlled by the -log option to a temporary file that it scans for the specified string(s). (The stream that is controlled by the -altlog option is not touched by the %CPLOGRC macro.)

Next, within the SAS session and after a call to the %CPSTART macro, set the macro variable CPLOGRC to the value REPORT or the value QUIETLY. Here's an example:

```
%let CPLOGRC=QUIETLY; * Scan the log and set the return code from %CPLOGRC;

%let CPLOGRC=REPORT; * Scan the log, set the return code, and print report ;
                     * (in both the -log and -altlog streams) on which ;
                     * strings were found.;
```

After these two steps, you can use one or more sequences of calls to the %CPLOGRC macro. For more information about sequences of calls, see the *opcode* parameter.

%CPLOGRC Examples

Example 1

This example checks the SAS log for error messages (messages that begin with "ERROR:") and terminates the SAS session if any error messages are found. The default value of the LOOKFOR= parameter is the string "ERROR:" and the default value of the WHEREAT= parameter is BEGINLINE.

```
%let CPLOGRC=REPORT;
* Specify that report is to be written to log at CHECK ;

%CPLOGRC(START);
*   Redirect SAS log for later scanning;

.  .  .  SAS programming statements here .  .  .

%CPLOGRC( CHECK, _rc=myrc);
*   Scan the log for ERROR: at BEGINLINE ;

%CPFAILIF (&myrc ne 0, code=999);
* If ERROR: found, terminate and set exit code;
* to 999 -check the SAS log for more details on error;
```

Example 2

This example performs the same tasks as Example 1 and then checks the SAS log for warning messages (messages that begin with "WARNING:"), error messages (messages that begin with "ERROR:"), and catastrophe messages (messages that use the string **catastrophe** anywhere in the message). Terminate the SAS session if any such messages are found, and set the exit code to distinguish the most serious reason for failure. Finally, if some other problem occurred, then terminate the SAS session and set the code to the highest return code that is found in all the calls to %CPLOGRC (including the calls that did not set the _RC= parameter).

```
%let CPLOGRC=REPORT;
* Specify that report is to be written to log at CHECK ;
```

```
%CPLOGRC( START);
*   Redirect SAS log for later scanning;

.  .  .  SAS programming statements here .  .  .

%CPLOGRC( CHECK, _rc=myrc);
*   Scan the log for ERROR: at BEGINLINE ;

%CPFAILIF (&myrc ne 0,  code=999);
* If ERROR: found, terminate and set exit code;
* to 999 -check the SAS log for more details on error;

    ...programming statements...
%CPLOGRC (CHECK,
          lookfor=WARNING: ERROR:  catastrophe,
          whereat=beginline  beginline  anywhere,
          _howmany=counts ) ;
* Scan the log for WARNING: at BEGINLINE ;
* and ERROR: at BEGINLINE ;
* and catastrophe anywhere on line ;

%CPFAILIF (%SCAN(&counts,3)>0, code=998);
*Fail if one or more catastrophe;

%CPFAILIF (%SCAN(&counts,2)>0, code=997);
*Fail if one or more errors;

%CPFAILIF (%SCAN(&counts,1)>0, code=996);
*Fail if one or more warnings;

%CPFAILIF(&CPLRMAX ne 0,  code=%CPLRMAX);
*Set exitcode to max retcode;

%CPLOGRC (STOP) ;
```

%CPLVLTRM

Trims or removes data from a table in the active PDB based on the age limit for that table in the PDB's data dictionary

%CPLVLTRM Overview

The %CPLVLTRM macro enables you to age out or "trim" the data in a specific level (DETAIL, DAY, WEEK, MONTH, or YEAR) of a table in the active PDB. The datetime range of data for the specified level and table in the active PDB is compared with the age limit for the same level and table in the PDB's data dictionary. If the datetime range of the data in the specified level and table in the active PDB is greater than the age limit in the same level and table in the data dictionary, then the data in the specified level and table in the active PDB is trimmed to match the age limit of the data in the same table in the data dictionary.

For example, in the active PDB, you might have data that spans 15 days in the day level of table XYZ. You might decide to change the limit to 10 days. To reclaim the space that is used by the unneeded 5 days of data, you could then run %CPLVLTRM on that PDB, table, and level. If the age limit for the day level in table XYZ in that PDB's data dictionary is 10 days, then 5 days of data would be trimmed from the day level of table XYZ in the active PDB. Note that if you do not perform the trim by using %CPLVLTRM, then the next process-and-reduce step does the trim. The %CPLVLTRM macro moves the trim earlier in time.

%CPLVLTRM Syntax

%CPLVLTRM(
LEVEL=DETAIL | DAY | WEEK | MONTH | YEAR
,**TABLE=***tablename*
<,**_RC=***macro-var-name*>);

Details

LEVEL=DETAIL | DAY | WEEK | MONTH | YEAR
specifies the level of the specified table (in the active PDB) from which data should be aged out or removed.

TABLE=*tablename*
specifies the name of the table (in the active PDB) that contains the data that you want to age out.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPLVLTRM Notes

CAUTION:
This macro should be used only under the direct supervision of SAS Technical Support.
You should always make a backup of your PDB before you use this macro. △

This macro ages out or removes data on one level of the specified table. Therefore, to age out data on more than one level of one table or on more than one table, you must submit the macro one time for each level of each table.

Do not use this macro if you are experiencing any problems with your PDB. Also, do not use this macro between a run of %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES and a run of %CPREDUCE (that is, after data is processed but before that same data is reduced).

Note: At the week, month, or year level, if AGELIMIT=*number*, then data is kept for the most recent partial week, month, or year and for the previous *number-1* full weeks, months, or years. In other words, AGELIMIT=*number* means up to but not including *number* full weeks, months, or years of data. △

%CPLVLTRM Example

This example trims the data at the detail level in table XTY70 in the active PDB.

```
%CPLVLTRM(level=DETAIL, table=XTY70);
```

%CPPDBOPT

Sets or copies PDB options

%CPPDBOPT Overview

The %CPPDBOPT macro enables you to set PDB options in the active PDB, to set PDB options directly in your site library (SITELIB), to copy PDB options between libraries, and/or to print PDB options in libraries.

%CPPDBOPT Syntax

%CPPDBOPT(
<**ARCDEV=***archive-device*>
<,**ARCENGIN=**D | T>
<,**ARCPARM=***archive-opts*>
<,**ARCPATH=***archive-path*>
<,**COPY=**DICTLIB | SITELIB | PGMLIB>
<,**CPACTCOL=***collector*>
<,**CPDSTIME=***filename-or-code*>
<,**CPGMTDEV=***hours-from-gmt*>
<,**CPSOW=***start-of-week*>
<,**CPXPDBNM=**PDB>
<,**FRI=***shifts*>
<,**MON=***shifts*>
<,**PRINT=**YES | NO>
<,**_RC=***macro-var-name*>
<,**SHIFTDSC=***shift-descriptions*>
<,**SAT=***shifts*>
<,**SUN=***shifts*>
<,**THU=***shifts*>
<,**TUE=***shifts*>

<,**TYPE=**DICTLIB | SITELIB | PGMLIB>
<,**WED=***shifts*>);

Details

ARCDEV=*archive-device*
specifies the name of the device where you want to save or write the archive libraries. The device name can be a maximum of eight characters long. For z/OS, use the value of the UNIT= parameter that you specify in your JCL or that you would specify in a SAS LIBNAME statement. For example, use ARCDEV=TAPE if, on your system, you would specify UNIT=TAPE in order to write to a tape drive. *This parameter is required if you are archiving data on z/OS.* For operating environments other than z/OS, the parameter is optional.

For UNIX or Windows environments, you would specify the server, device name, or drive, in a format appropriate for the active PDB's host. For example, on Windows you can specify **`arcdev=E:`**, and on UNIX you might specify a directory path or specific location on a server. The value that you specify for this parameter is concatenated with the value of the ARCPATH= parameter. The concatenated values identify the full name of the location where you want to archive data that is being read into the PDB. (If necessary, the %CPPDBOPT macro supplies a slash (UNIX) or backward slash (Windows) where the ARCDEV= and ARCPATH= values are concatenated.) For more information on the values for this parameter, see the following Archiving Path Table.

Note: If you select the archive type of DISK (ARCENGIN=D) and omit the ARCDEV= and ARCPATH= parameters, then the location of the archive libraries defaults to the path of the active PDB. △

Table 2.8 Archiving Pathname Summary

Platform	ARCENGIN	ARCDEV	ARCPATH	Result: Archive written to
z/OS	DISK	SYSDA	(blank)	"pdb-root" with regular SAS engine and UNIT=SYSDA libname parm
UNIX	DISK	(blank)	(blank)	"pdb-root" with regular SAS engine
Windows	DISK	(blank)	(blank)	"db-root" with regular SAS engine
z/OS	DISK	DISK	MY.ARCHIV	"MY.ARCHIV" with regular SAS engine and UNIT=DISK libname parm
z/OS	TAPE	CART	MY.ARCHIV	"MY.ARCHIV" with sequential engine and UNIT=CART libname parm (points to tape drive)

Platform	ARCENGIN	ARCDEV	ARCPATH	Result: Archive written to
UNIX	DISK	/usr/ pdb	/archive	"/usr/pdb/archive" with regular SAS engine
Windows	DISK	f:	archive	"f:\archive" with regular SAS engine

ARCENGIN=D | T
indicates the type of device to which you are archiving data. This is the device type for the device that is identified in the ARCDEV= parameter. *D* specifies DISK (the regular SAS engine); *T* specifies TAPE (the sequential engine).

This parameter is not required. The default value is D, Disk. For more about library specifications when ARCENGEN=T, see the ARCPARM= parameter. See also the ARCDEV= parameter.

ARCPARM=*archive-opts*
indicates additional options that you want to specify on the archive library's LIBNAME statement that is generated automatically each time that data is processed and archived. The value of this parameter can include additional SAS LIBNAME parameters that are necessary to allocate the SAS data library where the archive will be written. The value of this parameter can be a maximum of 200 characters long. *This parameter is not required.* The value of the ARCPARM= parameter might be case sensitive depending on the archiving mechanism that is used.

Typically, the ARCPARM= parameter is used if the ARCENGIN= parameter is set to T. For example, on z/OS, if ARCENGIN=T, you might want to use something like

```
ARCPARM='LABEL=(1,SL,,,RETPD=365)'
```

For more information about the SAS LIBNAME statement, see the SAS Companion documentation for the platform where the archive (set of archive libraries) is to be stored.

ARCPATH=*archive-path*
specifies the complete directory path (UNIX or Windows environments) or high-level qualifiers (z/OS) of the archive libraries, but not the name of the libraries. Each archive library name is created when the library is archived. The archive path is the location where you want to store the libraries, on the device that is specified with the ARCDEV= parameter. For UNIX and Windows environments, the pathname is a concatenation of the values that are specified on the ARCDEV= and ARCPATH= parameters. On z/OS, the pathname is the value that is specified by the ARCPATH= parameter.

The archive path can be a maximum of 200 characters in length. The *archive-path* is case sensitive in UNIX but not in z/OS or Windows environments. *This parameter is not required. The default value is the root-level path or high-level qualifiers of the PDB that contains the data that you are archiving.*

Note: If you select the archive type of DISK (ARCENGIN=D) and omit the ARCDEV= and ARCPATH= parameters, then the location of the archive files defaults to the path of the active PDB. For more information, see the ARCDEV= parameter. △

COPY=DICTLIB | SITELIB | PGMLIB
identifies the library (SITELIB, PGMLIB, or the active PDB's DICTLIB) from which to copy all PDB options. When you specify the COPY= parameter, all PDB options are copied from that library (the value of COPY=) to the library that

specified in the TYPE= parameter (the target library). Modifications are applied to the target library in this order:

1. PDB options are copied from the library that is specified in the COPY= parameter to the library that is specified in the TYPE= parameter.
2. For any options that you specify on the %CPPDBOPT macro, the new values are applied to those options in the target library (which is specified in the TYPE= parameter).

COPY=SITELIB copies options from your site library to the target library (specified via the TYPE= parameter). COPY=DICTLIB copies options from the active PDB's DICTLIB to the target library. COPY=PGMLIB copies the original default options that are shipped with SAS IT Resource Management to the target library.

This parameter is not required. If you do not specify the COPY= parameter, then for any parameters that you do not specify on %CPPDBOPT, those PDB options retain their current values in the target library.

CPACTCOL=*collector*
specifies the name of the collector that you want to use as the default collector for the active PDB.

This parameter is not required and the default is the current value, if you have previously set this parameter.

CPDSTIME=*filename-or-code*
specifies the code that defines the datetime range during which daylight saving time is in effect. The value of this parameter is used to identify and convert from daylight saving time to standard time. This is a SAS expression that you want %CSPROCES to use in order to adjust the DATETIME value for daylight saving time.

This parameter is used only for data that is collected by SunNet Manager on UNIX. This parameter is not required and the default action is to adjust for daylight saving time.

Staging code that is written for use with the Generic Collector Facility does not reference this PDB option.

CPGMTDEV=*hours-from-gmt*
specifies the number of hours' difference in the local time zone from Coordinated Universal Time. For example, the value for the eastern United States is five, and the value for Heidelberg, Germany, is negative one.

This parameter is used for data that is collected by SunNet Manager on UNIX and for PROBEX/TRAKKER data. This parameter is used for data that is collected by HP OpenView Performance Agent only if the Greenwich Mean Time is not in the data's header records. This parameter is not required.

Staging code that is written for use with the Generic Collector Facility does not reference this PDB option.

For additional information on Coordinated Universal Time, see the section "Handling Time-Stamps for Coordinated Universal Time" in the chapter "Administration: Working with Data" in the *SAS IT Resource Management User's Guide*

CPSOW=*start-of-week*
specifies the day of the week that you want to use as the starting day for each week, for the week level in the PDB. Valid values are **`Sunday`**, **`Monday`**, **`Tuesday`**, **`Wednesday`**, **`Thursday`**, **`Friday`**, and **`Saturday`**. You must use the day name and not a number, and you cannot abbreviate the day name. For example, if you want *start-of-week* to be Sunday, then you would specify *CPSOW=SUNDAY*. If you want *start-of-week* to be Monday, then you would specify *CPSOW=MONDAY*.

If you specify an invalid value, then the specification is ignored. If you do not specify this parameter, then the existing value is used from the target library to which you are copying PDB options.

CPXPDBNM=PDB
requests that additional views be created. These views cause the detail-level tables in the PDB to resemble MXG views. Also, when you specify **CPXPDBNM=PDB**, an additional libref is created that points to the library (in the PDB) where the MXG views are stored. This parameter is used only on z/OS.

This parameter is not required. If you specify CPXPDBNM= PDB, then you can later reverse this action by specifying a blank value (CPXPDBNM=) for this parameter.

SUN=*shifts*
MON=*shifts*
TUE=*shifts*
WED=*shifts*
THU=*shifts*
FRI=*shifts*
SAT=*shifts*
defines the shifts for each day of the week. These parameters enable you to associate a time of day with a specific shift (that is, what time of day is the 1st, 2nd, or 3rd shift) by assigning a single character code (such as 0, 1, 2, and so on) for each hour.

Shifts are defined as a 24-byte string, where each byte represents the one-byte SHIFT abbreviation on a 24-hour clock. For example, the string

```
FRI=000000000111111112222222
```

sets the following shift times on Fridays: shift "0" is from 0:00 to 8:59:59, shift "1" is from 09:00 to 16:59:59, and shift "2" is from 17:00 to 23:59:59.

PRINT=YES | NO
specifies whether or not to print the values of the PDB options that are in the library that is specified by the TYPE= parameter.

PRINT=NO
Values for PDB options are not printed.

PRINT=YES
If you specify PRINT=YES and TYPE=DICTLIB, then the values for PDB options are printed to the SAS log from the active PDB. If you specify PRINT=YES and TYPE=SITELIB, then the site's default values for PDB options are printed from the active site library. If you specify PRINT=YES and TYPE=PGMLIB, then the built-in default values for PDB options are printed from the program library.

The default is PRINT=YES.

If you are modifying options for a PDB and you specify **PRINT=YES**, then the values for PDB options are printed before and after the modifications are applied.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

SHIFTDSC=*shift-descriptions*

enables you to assign a shift label to the shift code, which was defined by the SUN=, MON=, ... parameters. This parameter specifies a list that consists of pairs of a SHIFT code and its description; the pairs are separated by one or more blanks. Here's an example:

```
SHIFTDSC="1"="Prime"
         "2"="NonPrime"
         ...
         "R"="Remaining"
```

where the (single) character in the first pair of quotation marks must be alphanumeric and the character string in the second pair of quotation marks can be a maximum length of 40 characters. If you wrap the description in double quotation marks, then you cannot use double quotation marks within the description, but you can use single quotation marks or parentheses. If you wrap the description in single quotation marks, then you cannot use single quotation marks within the description, but you can use double quotation marks or parentheses.

The following table presents an example of the shift codes and labels that you might use:

Table 2.9 Shift Labels

Shift Code	Shift Label
0	HOLIDAY
1	WEEKDAY
2	WEEKNIGHT
3	WEEKEND
4	Undefined
5	Undefined
6	Undefined

Note: Shift descriptions are series of text strings that are associated with shift codes. You can change these as needed; the text has no hidden meaning. You can set the description for SHIFT=2 to 'Holidays', or 'Holidays/Weekends', or even 'Jabberwocky', because the descriptions make no difference in the way the shift codes function. In particular, using the label HOLIDAY does not cause the corresponding code to be used for HOLIDAYS. The HSHIFT= parameter in the %CPHDAY macro specifies the code that is to be used for holidays. △

TYPE=DICTLIB | SITELIB | PGMLIB

identifies the library from which to print and (in the case of DICTLIB and SITELIB) the library in which to modify PDB options.

If you specify DICTLIB *(the default value)*, then any new PDB options that you specify are applied to the active PDB.

If you specify SITELIB, then any new PDB options that you specify are applied to the active site library.

If you specify PGMLIB, then you can print the built-in options but you cannot modify them.

If you specify the COPY= parameter and the TYPE= parameter, then the PDB options are copied from the library that is specified in the COPY= parameter (the from library) to the library that is specified in the TYPE= parameter (the target library). Modifications are applied to the target PDB in this order:

1. PDB options are copied from the library that is specified in the COPY= parameter to the library that is specified in the TYPE= parameter.
2. For any options that you specify on the %CPPDBOPT macro, the new values are applied to those options in the target library (which is specified in the TYPE= parameter).

If you do not specify the COPY= parameter (that is, if you are only setting options) and you do not specify all the parameters on this macro, then the existing values for the unspecified parameters in the target library are retained.

%CPPDBOPT Notes

The values of the PDB options in the PGMLIB library are as shipped with SAS IT Resource Management. The values of the PDB options in the PGMLIB library represent the built-in default values. Typically, access to the PDB options in the PGMLIB library is READONLY (UNIX and Windows) and SHR (on z/OS).

Depending on the history of the SITELIB library, there might or might not already be values of the PDB options in the SITELIB library. The values of the PDB options in the SITELIB library represent the site's preferences for default values. Access to the PDB options in the SITELIB library is controlled by the setting of the SITEACC= parameter in the %CPSTART macro. If you have WRITE or OLD access, then you can use this macro to copy all the values of PDB options from PGMLIB (use COPY=PGMLIB) to SITELIB (use TYPE=SITELIB) or to copy all the values of PDB options from the active PDB's DICTLIB (use COPY=DICTLIB) to SITELIB (use TYPE=SITELIB). You can also use this macro to directly set one or more values of the PDB options in the SITELIB library (use TYPE=SITELIB). If a call to this macro specifies a copy and also one or more direct sets, then the copy takes place first.

Note: The updates take place immediately. That is, macros later in the same job see the updated values in SITELIB, not the old values. △

When a PDB is created, the values of its PDB options are copied from SITELIB (if there are PDB options in SITELIB) or PGMLIB (if there are no PDB options in SITELIB). Access to the PDB options in the active PDB's DICTLIB library is controlled by the setting of the ACCESS= parameter (for UNIX and Windows) or DISP= parameter (for z/OS) in the %CPSTART macro. If you have WRITE or OLD access, then you can use this macro to copy all the values of PDB options from PGMLIB (use COPY=PGMLIB) to the active PDB's DICTLIB (use TYPE=DICTLIB) or to copy all the values of PDB options from SITELIB (use COPY=SITELIB) to the active PDB's DICTLIB (use TYPE=DICTLIB). You can also use this macro to directly set one or more values of the PDB options in the active PDB's DICTLIB library (use TYPE=DICTLIB). If a call to this macro specifies a copy and also one or more direct sets, then the copy takes place first.

Note: The updates take place immediately. That is, macros later in the same job see the updated values in the active PDB's DICTLIB, not the old values. △

For more information on archiving, see "Archive Overview" on page 667.

%CPPDBOPT Examples

Example 1

The following code overwrites all PDB options in the active PDB with the default options from SITELIB, and then sets the active PDB's start of week to Tuesday.

```
%CPPDBOPT(type=DICTLIB,copy=SITELIB,cpsow=Tuesday,_rc=retcode);
```

Example 2

The following code changes the specified PDB options in your active site library (SITELIB). All PDB options that you do not specify retain their existing values in the active site library.

```
%CPPDBOPT(type=SITELIB,
          otherpdboptions=value,
          ...
          otherpdboptions=value,_rc=retcode);
```

Example 3

The following code changes the specified PDB options in the active PDB. All options that you do not specify retain their existing values in the active PDB's DICTLIB.

```
%CPPDBOPT(type=DICTLIB,
          otherpdboptions=value,
          ...
          otherpdboptions=value,_rc=retcode);
```

Example 4

The following code copies the values for all PDB options in the active site library to the active PDB.

```
%CPPDBOPT(copy=SITELIB,type=DICTLIB,_rc=retcode);
```

Example 5

The following code copies all values for PDB options in the active PDB to the active site library (SITELIB).

```
%CPPDBOPT(copy=DICTLIB,type=SITELIB,_rc=retcode);
```

Example 6

The following code restores the original values for PDB options, as shipped with SAS IT Resource Management, from the master data dictionary in (PGMLIB) to the active site library (SITELIB).

```
%CPPDBOPT(copy=PGMLIB,type=SITELIB,_rc=retcode);
```

Example 7

The following code affects the active PDB and

- sets the start of the week to Sunday in the week level
- specifies the location of the archive
- assigns shift code 1 from 8:00 to 17:59 on weekdays, shift code 2 for the remaining times on weekdays, and shift code 3 on the weekend
- assigns the description "PRIME" to shift code 1
- specifies that the variable CPPDRC is to receive the value of the return code from this macro call.

```
%cppdbopt ( cpsow=Sunday
          , arcengin=disk
          , arcdev=c:
          ' arcpath=c:\temp\newpdb2
          ' sun=333333333333333333333333
          , mon=222222221111111111222222
          , tue=222222221111111111222222
          ' wed=222222221111111111222222
          , thu=222222221111111111222222
          , fri=222222221111111111222222
          , sat=333333333333333333333333
          ' shiftdesc= ''1''=''PRIME''
          , _rc=cppdrc
          );
%put CPPDBOPT return code is &cppdrc;
%cpfailif (&cppdrc ne 0, code=999);
```

Note: Some default values of the PDB properties are specified in the above call to make it self-documenting. By default, any changes that you specify take place in the active PDB immediately. △

Example 8

The following code shows %CPPDBOPT parameters that differ by platform:
For UNIX:

```
          .
          .
          , arcdev=c:
          , arcpath=c:/temp/newpdb2
          .
```

For Windows:

```
          .
          .
          , arcengin=disk
          , arcdev=c:
          ' arcpath=c:\temp\newpdb2
          .
```

For z/OS:

```
      .
      .
```

```
, arcdev=sysda
, arcpath=location.of.newpdb2
.
```

Note: If ARCENGIN=TAPE, you can use an ARCPARM similar to

```
, arcparm='LABEL=(1,SL,,,RETPD=365)'
```

△

%CPPKGCOL

Creates a package of the user-written collector-support entities for a given data collector or data source

%CPPKGCOL Overview

The %CPPKGCOL macro creates a package that contains entities for supporting a given data collector or data source. The collector-support entities include

- the table definitions and their variable definitions (required)
- the staging code (required)
- a call to the duplicate-data-checking macro %CPDUPCHK (optional)
- the informats (optional)
- the exit routines (optional)
- the report definitions (optional)
- the exception rule definitions (optional)
- the formats (optional)
- the documentation (optional, but strongly recommended).

The package that is created is in the format of a SAS transport file. If you want to be reminded of the contents of the package, call the %CPRPTPKG macro. If you want to install the contents of the package, use a mechanism such as binary FTP to transport the package (if necessary) to the system where it is to be installed, and then call the %CPINSPKG macro.

%CPPKGCOL Syntax

%CPPKGCOL(
COLLECTR=*collector-name*
,**DELIM=**REQUIRED | NOTVALID | OPTIONAL
,**PACKFILE=***fileref*
,**TOOLNM=***tool-name*
<,**DUPMODE=**REQUIRED | NOTVALID | OPTIONAL>
<,**MACHINE=**REQUIRED | NOTVALID | OPTIONAL>
<,**ONETABLE=**YES | NO>
<,**OPSYS=***list*>
<,**RAWDATA=**REQUIRED | NOTVALID | OPTIONAL>
<,**RAWDIR=**YES | NO>
<,**_RC=***macro-var-name*>
<,**TABLIST=**REQUIRED | NOTVALID | OPTIONAL>);

Details

COLLECTR=*collector-name*
specifies name of the collector for which the collector-support entities are being packaged. The name should start with the letter U and contain letters and numbers. The maximum length of the name is 8 characters. The name must be different from any of the names listed in the COLLECTR= description on "%CPPROCES" on page 154.

The package is "tagged" with the value of the COLLECTR= parameter and the value of the TOOLNM= parameter. After installation of the package, the installed collector-support entities are also "tagged" with the value of the COLLECTR= parameter and the value of the TOOLNM= parameter. Thus, the collector name and tool name enable the software to find the appropriate set of collector-support entities to use.

DELIM=REQUIRED | NOTVALID | OPTIONAL
specifies a value that is available for use by the Process Wizard, if a user processes data from this data collector or data source by using the Process Wizard. The Process Wizard uses the value to determine whether to display the window named Delimiter and to determine how to control the window's behavior. (The window has one function, which is to enable the user to specify one or more delimiters that are used to separate values in the raw data.)

This parameter is optional. The default value is NOTVALID.

The valid values are

- REQUIRED, which means that the Process Wizard is to display the window and is not to activate the Next button until the user specifies at least one delimiter
- NOTVALID, which means that the Process Wizard is not to display the window
- OPTIONAL, which means that the Process Wizard is to display the window and is to activate the Next button whether or not the user specifies a delimiter.

  If you specify OPTIONAL and the user specifies one or more delimiters, then the delimiters are used as specified by the staging code. The staging code can use the user-specified delimiter(s) with or instead of the default delimiter.

  If you specify OPTIONAL and the user does not specify a delimiter, then the staging code needs to have a default delimiter.

For more information about how the staging code obtains the user-specified delimiter(s) from the Process Wizard, see the topic "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

PACKFILE=*fileref*
specifies a fileref. The fileref points to the external file where the package will be stored (if the package is being created) or is stored (if the package is being reported on or installed). The fileref must already exist by the time that the macro is called.

For example, suppose that you are creating in Windows a package of collector-supported entities and you want to use a fileref of **`pkgfile`**. Before your call to the %CPPKGCOL macro, you must have a SAS FILENAME statement similar to this one:

```
FILENAME pkgfile 'c:\ucollpkg.xpt' ;
```

where **`c:\ucollpkg.xpt`** is the location where the package is to be written.

Similarly, suppose that you are reporting on the same package, which has been FTP'd in binary mode to UNIX, and you want to use a fileref of **unixpack**. Before your call to the %CPRPTPKG macro, you must have a SAS FILENAME statement similar to this one:

```
FILENAME unixpack '\usr\itsv\ucoll\ucollpkg.xpt' ;
```

where **\usr\itsv\ucoll\ucollpkg.xpt** is the location of the package.

TOOLNM=*tool-name*
specifies the value that you will use when you process data from this data collector or data source. The typical value is SASDS.

The package is "tagged" with the value of the COLLECTR= parameter and the value of the TOOLNM= parameter. After installation of the package, the installed collector-support entities are also "tagged" with the value of the COLLECTR= parameter and the value of the TOOLNM= parameter. Thus, the collector name and tool name enable the software to find the appropriate set of collector-support entities to use.

DUPMODE= REQUIRED | NOTVALID | OPTIONAL
specifies a value that is available for use by the Process Wizard, if a user processes data from this data collector or data source by using the Process Wizard. The Process Wizard uses the value to determine whether to display the window named Specify Duplicate Checking when the option Duplicate Filtering is selected from the Advanced Options window. (The Specify Duplicate Checking window has one function. The user selects Filter Duplicate Data if duplicate data is to be removed by the duplicate-data-checking macros. The user does not select Filter Duplicate Data if the data is not to be checked for duplicates.)

Use of this parameter is optional. The default value of this parameter is NOTVALID.

The valid values are

- REQUIRED, which has the same meaning as OPTIONAL for the DUPMODE= parameter.
- NOTVALID, which indicates that the window is not to be displayed.

  If you set the value of the RAWDATA= parameter to NOTVALID, the Process Wizard uses the (default) setting DUPMODE=INACTIVE on the call to the %CPPROCES macro, and the staging code ignores the call, if there is one, to the %CPDUPCHK macro.
- OPTIONAL, which indicates that the window is to be displayed.

  If you set the value of the DUPMODE= parameter to OPTIONAL and the user selects Filter Duplicate Data, the Process Wizard uses the setting DUPMODE=DISCARD on the call to the %CPPROCES macro, and the staging code obtains the call to the %CPDUPCHK macro from the location where the %CPINSPKG macro installed it.

  If you set the value of the DUPMODE= parameter to OPTIONAL and the user does not select Filter Duplicate Data, the Process Wizard uses the (default) setting DUPMODE=INACTIVE on the call to the %CPPROCES macro, and the staging code ignores the call to the %CPDUPCHK macro.

For more information about how the staging code obtains the user's duplicate-data-checking choice from the Process Wizard, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

MACHINE=REQUIRED | NOTVALID | OPTIONAL
specifies a value that will be used by the Process Wizard to determine whether to display the window named Specify Machine and to determine how to control the

window's behavior. (This window has one function, which is to enable the user to specify the name of one machine.)

Use of this parameter is optional. The default value of this parameter is NOTVALID.

The valid values are

- REQUIRED, which indicates that the window is to be displayed and that the user must specify a machine name before the Process Wizard will activate the Next button.

  If you set the value of the MACHINE= parameter to REQUIRED, the name that the user specifies is available for the value of the variable MACHINE, as if the value had been in the raw data.

- NOTVALID, which indicates that the window is not to be displayed.

  If you set the value of the MACHINE= parameter to NOTVALID, the Process Wizard does not display the window. The value of the variable MACHINE must be in the raw data or must be provided by the staging code.

- OPTIONAL, which indicates that the window is to be displayed, but that the Next button is to be activated whether or not the user specifies the name of a machine.

  If you set the value of the MACHINE= parameter to OPTIONAL and the user specifies a name, that name is available for the value of the variable MACHINE, as if the value had been in the raw data.

  If you set the value of the MACHINE= parameter to OPTIONAL and the user does not specify a name, then the value of the variable MACHINE must be in the raw data or a default value must be provided by the staging code.

For more information about how the staging code obtains the user-specified machine name from the Process Wizard, see the topic "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

ONETABLE=YES | NO

specifies a value that is available for use by the Process Wizard, if a user processes data from this data collector or data source by using the Process Wizard. The Process Wizard uses the value to determine whether to permit the user to select only one table or multiple tables.

The use of this parameter is optional. The default value is NO.

The valid values are

- YES, which indicates that only one table can be selected in the Process Wizard
- NO, which indicates that one or more tables can be selected in the Process Wizard.

For more information, see the TABLIST= parameter.

OPSYS=*list*

specifies a list of one or more keywords. Each keyword identifies an operating system on which you checked that the collector-support entities in this package work correctly. If you specify more than one keyword, use white space between adjacent keywords.

This parameter is optional. The default value is the keyword that identifies the operating system on which the package is being created.

The following keywords are valid:

- MVS (for z/OS)
- WIN (for Windows)
- UNIX.

Note: Depending on the work that is being done in your exit routines and/or staging code, you may need to have the code test for the (automatically set) value of the macro variable CPOPSYS and branch to handle something in an operating system-dependent way.

Setting the OPSYS= parameter on the %CPPKGCOL macro does not ensure that the code will run on those operating systems. It confirms that you have checked that the code works on those operating systems.

For more information about the macro variable CPOPSYS, see "Global and Local Macro Variables" on page 6. △

RAWDATA= REQUIRED | NOTVALID | OPTIONAL
specifies a value that is available for use by the Process Wizard, if a user processes data from this data collector or data source by using the Process Wizard. The Process Wizard uses the value to determine whether to display the window named Specify the Location of the Raw Data and to determine how to control the window's behavior. (This window has two functions. The first is to specify whether the input data is in a single file, is in a directory that contains multiple files of raw data (on UNIX or Windows, or the z/OS UNIX File System on z/OS), or is in a PDS that contains multiple members of raw data (on z/OS). A single file is the default. The second function is to specify the location of that single file or that directory or PDS. Note that the value of the RAWDIR= parameter affects the window's first function and that the value of the RAWDATA= parameter affects the window's second function.)

Use of this parameter is optional. The default value of this parameter is OPTIONAL.

The valid values are

- REQUIRED, which indicates that the window is to be displayed and that the user must provide the location of the raw data before the Process Wizard enables the Next button.

  If you set the value of the RAWDATA= parameter to REQUIRED, the Process Wizard obtains the location of the raw data from the user and creates a fileref named RAWDATA that points to that location. The staging code can use the fileref RAWDATA to access the data.

  Note that the value of the RAWDIR= parameter may affect whether the location can be a directory (of files containing raw data, on UNIX or Windows, or in a z/OS UNIX File System area on z/OS), a PDS (of members containing raw data, on z/OS), or only a single file.
- NOTVALID, which indicates that the window is not to be displayed.

  If you set the value of the RAWDATA= parameter to NOTVALID, the Process Wizard does not obtain the location of the raw data from the user. The staging code needs to know where the location is. The staging code should use a fileref named RAWDATA to point to that location.
- OPTIONAL, which indicates that the window is to be displayed, but that the Next button is to be activated whether or not the user provides the location of the raw data.

  If you set the value of the RAWDATA= parameter to OPTIONAL and the Process Wizard obtains the location of the raw data from the user, the Process Wizard creates a fileref named RAWDATA that points to that location (or uses an existing fileref named RAWDATA and points it to that location). The staging code can use the fileref RAWDATA to access the data.

  If you set the value of the RAWDATA= parameter to OPTIONAL and the Process Wizard does not obtain the location of the raw data from the user, the staging code needs to have created a fileref named RAWDATA that points to a default location.

Note that the value of the RAWDIR= parameter may affect whether the location can be a directory (of files containing raw data, on UNIX or Windows, or in a z/OS UNIX File System area on z/OS), a PDS (of members containing raw data, on z/OS), or only a single file.

RAWDIR=YES | NO
specifies a value that is available for use by the Process Wizard, if a user processes data from this data collector or data source by using the Process Wizard. The Process Wizard uses the value to determine whether to permit the user to select a directory (on UNIX or Windows, or in a z/OS UNIX File System area on z/OS) or a PDS (on z/OS) as the location of the input data.

This parameter is optional. The default value is NO.

The valid values are

- YES, which indicates that the user can specify that the raw data is in files in a directory (on UNIX or Windows, or in a z/OS UNIX File System area on z/OS) or members in a PDS (on z/OS). The user can instead specify that the raw data is in a single file.
- NO, which indicates that the user cannot specify that the raw data is in files in a directory (on UNIX or Windows, or in a z/OS UNIX File System area on z/OS) or members in a PDS (on z/OS). The raw data must be in a single file.

For more information on RAWDIR=, see the RAWDATA= parameter.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

TABLIST=REQUIRED | NOTVALID | OPTIONAL
specifies a value that is available for use by the Process Wizard, if a user processes data from this data collector or data source by using the Process Wizard. The Process Wizard uses the value to determine whether to display the window named Specify Tables to Be Processed and to determine how how to control the window's behavior. (This window has two functions. The first function is to specify whether data from all tables is to be processed or whether data from selected tables is to be processed. The second function is to enable the user to select one or more tables, if that option was selected.)

Use of this parameter is optional. The default value of this parameter is OPTIONAL.

The valid values are as follows.

- REQUIRED has the same meaning as OPTIONAL for the TABLIST= parameter.
- NOTVALID indicates that the window is not to be displayed.

  If you set the value of the RAWDATA= parameter to NOTVALID, the Process Wizard will not obtain the table list from the user. So the Process Wizard will call %CPPROCES without specifying the tablist parameter. The %CPPROCES macro, when called without the tablist parameter, will use all of the tables that are "tagged" for the collector name and tool name and that have KEPT=YES.
- OPTIONAL indicates that the window is to be displayed, but that the Next button is to be activated whether or not the user selects anything in the window.

  If you set the value of the TABLIST= parameter to OPTIONAL and the user does not make any selections in the window, then All Tables is the default. The Process Wizard will process data for one or more tables "tagged" by the combination COLLECTR=*collector-name* and TOOLNM=*tool-name*. (Whether only the first such table or all such tables are used is controlled by the value of the ONETABLE= parameter.)

  If you set the value of the TABLIST= parameter to OPTIONAL and the user selects Selected Tables, then the Process Wizard will process data for the selected tables if the user does select one or more tables. If the user does not select one or more tables, then the Process Wizard will process data for all tables with KEPT=YES in the set of tables "tagged" by the collector name and tool name. (Whether only the first such table or all such tables are used is controlled by the value of the ONETABLE= parameter.)

%CPPKGCOL Notes

With one exception, the entities that are to be placed in the package must reside in the PDB that is active when this macro is called. (The exception is that if you are including in the package a report definition that is based on the %CPSRCRPT macro, the source code to which the rptname parameter points does not need to reside in the PDB. However, the source code does need to be accessible when the package is created.) Typically, the PDB that is active at the time that the package is created is a PDB that you created for this purpose and, thus, contains entities only for the data collector or data source for which you are creating the package.

The collector-support entities that are to be packaged must reside in certain locations in the active PDB for the %CPPKGCOL macro to copy them into the package.

- *Table and variable definitions:* You must place the table definitions and their variable definitions (including any definitions for derived variables and formula variables) in the data dictionary of the active PDB (DICTLIB).

  The active PDB must have at least one table definition for this collector-name and tool-name combination. If it does not, the macro stops.

  Note: You can create the table and variable definitions in batch mode, by using the %CPDDUTL macro and its CREATE TABLE, CREATE VARIABLE, UPDATE TABLE, and UPDATE VARIABLE control statements. Remember to specify the COLLECTOR=*collector-name* and TOOLNAME=*tool-name* settings in the CREATE TABLE or UPDATE TABLE control statement. For more information, see the "Using the %CPDDUTL Macro" on page 567.

  Or you can create the table and variable definitions interactively, by using a SAS IT Resource Management server GUI. Remember to specify the collector name and tool name values for each table. △

Note: Also see the staging code information below for an additional table property to specify for each table. △

- *Staging code:* You must place the staging code in one of the following two locations:
  - Use ADMIN.*collector-name.tool-name*.SOURCE, if the code is written by using the SAS DATA step and/or SAS procedures.
  - Use ADMIN.*collector-name.tool-name*.SCL, if the code is written by using SCL.

  Staging code must be in one of these locations. If it is not, the macro stops.

  Later, when your staging code runs, it will produce one or more SAS data sets/views. For each table in the PDB, there will be a data set/view that holds the staged data for that table. The staging code should produce each of these data sets/views in the location COLLECT.*external-table-name*, where *external-table-name* is the external name that is specified for the corresponding table in the table definition.

  Note: In batch mode, specify the EXTNAME= parameter in the CREATE TABLE or UPDATE TABLE control statement for each table. Interactively, in a SAS IT Resource Management server GUI, specify the External Name property of each table. △
- *Duplicate-data-checking macros:* If you want to use duplicate-data checking as implemented with the duplicate-data-checking macros, you must place a call to the %CPDUPCHK macro in ADMIN.*collector-name*.CPDUPCHK.SOURCE, and the call must have appropriate values for the %CPDUPCHK parameters.

  If no call is in that location, the macro continues.
- *Informats:* You may provide informats for use when the staging code reads the raw data. You must place each informat in ADMIN.CPFMTS.

  If no informats are in that location, the macro continues.
- *Exit routines:* You may provide exit routines for use in the process task. You must place each exit routine in
  - ADMIN.EXITSRC.*exitpoint*.SOURCE

  where *exitpoint* is the name of the exit point at which you want that exit routine to execute.

  If no exit routines are in those locations, the macro continues.
- *Report definitions:* You may provide one or more report definitions. If you generate the report definition(s) by using a SAS IT Resource Management server GUI, save the report definition(s) to the following folder:
  - ADMIN.ITSVRPT

  Note: If you use the %CPSRCRPT macro to define a report, the source code that is specified by the rptname parameter (in the %CPSRCRPT call) does not need to be in ADMIN.ITSVRPT. However, it should be in a location that will be accessible when the %CPPKGCOL macro is called. △
- *Rule definitions:* You may provide rule definitions (exception rules). You must place each rule definition in the folder named
  - ADMIN.CPXRULE

  If no rules are in this location, the macro continues.
- *Formats:* You may provide formats. You must place each format in
  - ADMIN.CPFMTS

  If no formats are in this location, the macro continues.

- *Documentation:* You may provide documentation that is specific to this collector or data source. You must place the text (in HTML format) in
  - ADMIN.*collector-name*.UHTMDFLT.SOURCE (for the main text)
  - ADMIN.*collector-name*.UHTM*.SOURCE, where * is one or more values other than DFLT (for any other chunks of text to which the main text links).

  Note: In the text, use a maximum line length of 68 characters. (Using a line length that is greater than 68 causes problems when installing a package on z/OS.) △

  If no text is in the location for the main text, the macro continues. However, it is strongly recommended that you have documentation for the collector-support entities in the package and for any other relevant aspects of working with this data collector or data source.

  Note: If you generate a report on this package (by calling %CPRPTPKG or %CPINSPKG), some general documentation is created, which includes a list of the collector-support entities that are in the package. This general documentation automatically links to the documentation that you specified for the main text and to any other chunks to which the main text links. △

After the package is installed, the user can use the entities interactively or in batch mode. For example, the user can process data by using the Process Wizard or by using the %CPPROCES macro.

- *Process Wizard:* If the user is processing data by using the Process Wizard, the values of the parameters DUPMODE=, MACHINE=, ONETABLE=, RAWDATA=, RAWDIR=, and TABLIST= are available to the Process Wizard so that the wizard can display the appropriate windows to the user and, as a result of the user's actions in those windows, process the data.

  The %CPPKGCOL macro assumes that the values of these parameters are compatible with the entities in the package. For example, there is no purpose in having the Process Wizard request information from the user about duplicate data checking if the package does not contain a call to the %CPDUPCHK macro. Thus, you must ensure that the behavior that you request of the Process Wizard matches the entities that the Process Wizard will attempt to use.

- *%CPPROCES macro:* If the user is processing data by using the %CPPROCES macro, the values of the parameters DUPMODE=, MACHINE=, ONETABLE=, RAWDATA=, RAWDIR=, and TABLIST= are irrelevant.

  The %CPPKGCOL macro assumes that the documentation provided in the package is compatible with the entities in the package. For example, there is no purpose in telling the user how to use the DUPMODE= parameter on the %CPPROCES macro if the package does not contain a call to the %CPDUPCHK macro. Thus, you must ensure that the documentation that you provide in the package matches the entities that you provide in the package.

Note that the staging code must be compatible with both interactive and batch mode. For example, if the Process Wizard or the %CPPROCES macro permits the user not to specify some necessary value, the staging code must provide a default value.

For more information about collectors and collector support, see the topic "Setting Up the Server" in the chapter "Setup: Introduction" in the *SAS IT Resource Management User's Guide*. For more information about packaging and installing collector support, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*, "%CPINSPKG" on page 124, and "%CPRPTPKG" on page 177.

%CPPKGCOL Examples

Example 1

```
FILENAME collpack 'c:\packfile.xpt' ;

%CPPKGCOL(packfile=collpack
        , collectr=UCOLL
        , toolnm=SASDS
        , _rc=pkgrc) ;

%put %CPPKGCOL return code is &pkgrc ;
```

This example creates a fileref named **collpack**. The fileref points to the file **c:\packfile.xpt**.

Next, the example packages a set of collector-support entities into the file to which the fileref points. The package is for a collector that is named **UCOLL** and a tool name of **SASDS**.

Then, the example writes the return code from the packaging operation.

After you install the package, users will be able to call %CPPROCES as follows:

```
%CPPROCES(...COLLECTR=UCOLL,TOOLNM=SASDS,...) ;
```

Example 2

This example packages user-written collector-support entities for a collector that is named **UCOLL** and a tool name of **SASDS**. The **OPSYS=WIN UNIX MVS** setting confirms that the collector-support entities have been tested on Windows, UNIX, and z/OS.

The values of the RAWDIR= and RAWDATA= parameters are provided for the use of the Process Wizard, in case any user processes data by using the Process Wizard after the package is installed. The RAWDIR=YES setting indicates that the Process Wizard must permit the user to specify that the raw data is in a directory that may contain multiple files of raw data. (The user may also specify that the raw data is in a single file.) The RAWDATA=REQUIRED parameter indicates that the Process Wizard must display a window that enables the user to specify the location of the directory (or single file) and that the Process Wizard must not go ahead to the next window until the location is specified. The other parameters that provide information for the Process Wizard take their default values.

```
FILENAME collpack 'c:\packfile.xpt' ;

%CPPKGCOL(packfile=collpack
        , collectr=UCOLL
        , toolnm=SASDS
        , opsys=WIN UNIX MVS
        , rawdir=YES,
        , rawdata=REQUIRED,
        , _rc=pkgrc) ;

%put %CPPKGCOL return code is &pkgrc ;
```

After you install the package, users will be able to use calls to %CPPROCES that are similar to this one:

```
%CPPROCES(...,COLLECTR=UCOLL,TOOLNM=SASDS,RAWDATA=location-of-directory,...) ;
```

%CPPROCES

Processes data into a PDB

%CPPROCES Overview

A call to the %CPPROCES macro processes raw or staged data into the active PDB, usually into the detail level.

A call to the %CPPROCES macro is typically followed by a call to the %CPREDUCE macro to summarize that data into the day, week, month, and year levels of the active PDB.

%CPPROCES Syntax

%CPPROCES(
<*tablist*>
,**GENLIB=***libref*
<,**COLLECTR=***collector-name*>
<,**DELIM=**'delim-chars'>
<,**DUPMODE=**INACTIVE | FORCE | DISCARD | TERMINATE>
<,**EXITSRC=***exit-source*>
<,**GROUPDIR=***location-of-group-directory*>
<,**RAWDATA=***filename*>
<,**_RC=***macro-var-name*>
<,**TOOLNM=***tool-name*>);

Details

tablist
lists the names of the tables into which the logged data should be processed. *This positional parameter is not required*. If you do not specify this parameter, then use a comma as a placeholder. If you specify more than one table, then use blanks to separate the tables in the list.

If you specify one or more tables, then the data in these tables is processed into the PDB.

If you do not specify a list of tables, then the data is processed into the PDB for all tables that meet the following criteria:

- tables that are associated with the collector that is specified in the COLLECTR= parameter
- tables that have a status of KEPT=Y
- tables that have at least one variable at detail level with a KEPT status of YES.

If you specify a table in the list and the table is not found in the active PDB, then the table definition that is supplied is added to the active PDB from the master data dictionary.

GENLIB=*libref*
specifies the *libref* for the SAS library that contains the staged data that is to be processed into the PDB. The staged data has been created by user-provided software and is to be processed into a PDB by using COLLECTR=GENERIC. The

data may be a SAS data set, a SAS DATA step view, a SAS SQL view, or a SAS/ACCESS view. For each table, the name of the data set, DATA step view, SQL view, or SAS/ACCESS view that has staged data for the table is specified by the table's external name parameter.

Use this parameter to specify the input data set or view for the SAS IT Resource Management process macro that you use to process your data, but only if you specify COLLECTR=GENERIC and TOOLNM=SASDS. If you specify this parameter for the %CPPROCES macro, then do not specify the RAWDATA= parameter. If you specify the GENLIB= parameter for the %CMPROCES macro or the %CSPROCES macro or the %CWPROCES macro, then do not specify the (positional) *rawdata* parameter.

COLLECTR=*collector-name*
specifies the name of the data collector or data source that provides the raw data. The value that you specify for the COLLECTR= parameter depends on which process macro you use. The following example uses %CSPROCES with the SUNETMGR collector:

```
%CSPROCES( ...collectr=sunetmgr,...);
```

Also, the TOOLNM= parameter may be required depending on which value you specify for the COLLECTR= parameter. The following example uses %CWPROCES with the WEBLOG collector:

```
%CWPROCES( ...collectr=weblog, toolnm=clf,...);
```

See the following COLLECTR - TOOLNM Parameter Values table for valid values of the COLLECTR= parameter and the TOOLNM= parameter. In the COLLECTR= Value and TOOLNM= Value columns of this table, the following process macro abbreviations apply:

%CM for %CMPROCES

%CP for %CPPROCES

%CS for %CSPROCES

%CW for %CWPROCES.

Table 2.10 COLLECTR - TOOLNM Parameter Values

COLLECTR= Value	Collector Description	TOOLNM= Value
GENERIC for %CM, %CP, %CS, and %CW	data collector or data source for which SAS IT Resource Management does not supply table definitions and the user has not installed table definitions	SASDS for %CM SASDS or CHARDELIM for %CP, %CS, and %CW
ACCUNX for %CP	UNIX accounting	SASDS
DCOLLECT for %CM	IBM DFSMS Data Collection Facility	MXG
EREP for %CM	IBM SYS1.LOGREC Error Recording	MXG

COLLECTR= Value	Collector Description	TOOLNM= Value
ETEWATCH for %CP	End-To-End Watch (transaction response measurement) by Candle	ETE12 (ETEWatch 1.2 from Candle) ETE13 (ETEWatch 1.3 from Candle) EBA (EBA logs from Candle)
HP-MWA for %CS and %CW	HP MeasureWare	MWA-BE or MWA-LE (See Note 6 following the table)
HP-OV for %CS and %CW	IBM Tivoli NetView or HP Network Node Manager	not applicable
HP-OV-C for %CS	IBM Tivoli NetView or HP Network Node Manager + Coerce	not applicable
HP-OVPA for %CS and %CW	HP OpenView Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
HP-PCS for %CS and %CW	HP Performance Collection Software	MWA-BE or MWA-LE (See Note 6 following the table)
HP-PCS for %CP	HP OpenView Reporter	SASACCESS
HP-VPPA for %CS and %CW	HP VantagePoint Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
IMF for %CM	Boole & Babbage IMF from BMC	MXG
LSPW for %CS and %CW	Landmark Performance Works	CHARDELIM
NETCONV for %CP	Cisco IOS NetFlow	NETFLOW
NTSMF for %CM and %CP	Windows with Demand Technology's NTSMF product	MXG for %CM SASDS for %CP

COLLECTR= Value	Collector Description	TOOLNM= Value
PATROL for %CP	BMC Patrol	SASDS
PROBEX for %CS	Probe/X by Landmark in SunNetMgr format	not applicable
ROLMPBX for %CP	Rolm PBX	SASDS
SAPR3 for %CP	SAP R/3	SASADAPT (SAS IT Management Adapter for SAP) STATBIN (Statistics File) STATRFC (Using SAP R/3 API) (See Note 7 following the table)
SAR for %CP	UNIX sar command (system activity reporting)	SASDS
SITESCOP for %CP	SiteScope by Mercury Interactive	SASDS
SMF for %CM and %CP	IBM z/OS System Management Facilities	MXG
SPECTRUM for %CS	SPECTRUM by Aprisma	not applicable
SUNETMGR for %CS	SunNet Manager and Enterprise Manager	not applicable
TMON2CIC for %CM	Landmark “The Monitor for CICS/ESA 2”	MXG
TMONCIC8 for %CM	Landmark “The Monitor for CICS” (older)	MXG
TMONCICS for %CM	Landmark “The Monitor for CICS” V 1.3 and later	MXG

COLLECTR= Value	Collector Description	TOOLNM= Value
TMONDB2 for %CM	Landmark "The Monitor for DB2"	MXG
TMS for %CM	CA-TMS (Release 5 or later)	MXG
TPF for %CM	IBM Transaction Processing Facility	MXG
TRAKKER for %CS	Trakker by Concord Communications	not applicable
VISULIZR for %CP	Visualizer by BMC Software	SASACCESS
VMMON for %CM	VM Monitor	MXG
WEBLOG for %CP	Web server log	CLF (Common Log Format) ELF (Extended Log Format) IISORIG (Microsoft IIS original Weblog format) WEBETE (ETEWatch Web browser logs from Candle) WEBEBA (EBA logs from Candle)

1 *For COLLECTR=GENERIC:* For more information about using the Generic Collector Facility, see the chapter "Setup Case 3" in the *SAS IT Resource Management User's Guide* and the chapter "Setup Case 4" in the *SAS IT Resource Management User's Guide*, and the chapter "Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

   For more information about installing user-written or consultant-written table definitions, see "%CPPKGCOL" on page 144, "%CPRPTPKG" on page 177, and "%CPINSPKG" on page 124.

2 *For %CMPROCES:* The COLLECTR= parameter is required. The TOOLNM= parameter is also required.

3 *For %CPPROCES:* The COLLECTR= parameter is optional. If the parameter is not specified, then the default is COLLECTR=GENERIC. The TOOLNM= parameter is also optional. If the parameter is not specified, then the default is TOOLNM=SASDS.

4 *For %CSPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

5 *For %CWPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

6 *For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA:* The TOOLNM= parameter is used only when you use a PIPE command for the raw data.

For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA, the TOOLNM= value is used to identify the endian type of the data. For example, data from Windows platforms is little-endian and data from most UNIX platforms is big-endian.

Specify TOOLNM=MWA-BE if the incoming data is big-endian (BE). Specify TOOLNM=MWA-LE if the incoming data is little-endian (LE).

If you use a PIPE command for the raw data and you do not specify the TOOLNM= parameter, SAS IT Resource Management attempts to determine the endian type of the incoming data. If the attempt is successful, the processing step continues. If the attempt is not successful, the processing step terminates with a message.

Note: HP OpenView Performance Agent (HP-OVPA) is the latest name for these products: HP-PCS, HP-MWA, and HP-VPPA. △

7 *SAP:*

- *For SAP Release 4.6D and earlier, use TOOLNM=STATBIN or TOOLNM=STATRFC.*

  You must set one macro variable before you run %CPPROCES. Additionally, there are four other optional variables that you might need to set, depending on the data that you are processing.

  The required macro variable is

  - SAPVER - the SAP R/3 version.

  The optional macro variables are

  - SAPSYSNR - the SAP R/3 system number
  - SAPSYSNM - the SAP R/3 system name
  - SAPHOST - the SAP R/3 application server name
  - SAPPLAT - the platform on which the SAP STAT file was generated.

  You need to specify the SAPPLAT macro variable only if you are processing SAP STAT data that was generated in a Windows operating environment. The default value for the SAPPLAT macro variable enables you to read data that is generated in both UNIX and z/OS operating environments. However, if you read data that was generated in a Windows operating environment, then set SAPPLAT to a value of WIN. (For more information, see the examples for the %CPPROCES macro.)

  You must specify the SAPSYSNM, SAPHOST, and SAPSYSNR macro variables only if you use the SAP R/3 collector and only if you process a single data file.

  To populate these macro variables automatically, see the instructions in the RAWDATA= parameter for the %CPPROCES macro.

- *For SAP releases later than 4.6D, use TOOLNM=SASADAPT.*

  You can also use TOOLNM=SASADAPT with some earlier releases of SAP.

8 *If you are working with a Candle collector:*

- Use COLLECTR=ETEWATCH, TOOLNM=ETE12 or ETE13 or EBA when application and transaction analysis is desired.

- Use COLLECTR=WEBLOG, TOOLNM=WEBETE or WEBEBA when processing WebBrowser behavior module data or EBA data for which a hierarchical page analysis (for example, jobs —> Cary —> R&D) is desired.

9 *For COLLECTR=SMF on UNIX or Windows:* See "Example 8: Processing SMF data on UNIX or Windows" in the %CPPROCES Examples.

10 Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

DELIM=*'delim_chars'*
specifies the character(s) that is used as a value separator in the character-delimited data. The delimiter character or list of delimiter characters must be enclosed in matched single or double quotation marks. Do not use a blank space to separate the items in the list. Specify a blank space in the list only if a blank space is one of the delimiters in your file.

If you do not specify a delimiter and you are processing character-delimited data, then the default value, a blank, is used.

Note: This parameter takes effect if TOOLNM=CHARDELIM or if COLLECTR=NTSMF.

DUPMODE=INACTIVE | FORCE | DISCARD | TERMINATE
specifies how duplicate-data checking is to be handled. *The default value is INACTIVE*.

If you *do not* want to use duplicate-data checking, set DUPMODE=INACTIVE (and, if COLLECTR=GENERIC, check that the staging code does not call the duplicate-data-checking macros).

INACTIVE
does not review the incoming data. Therefore, it allows duplicate data, if there is any, to be processed into the active PDB.

If you *do* want to use duplicate-data checking, set DUPMODE=FORCE, DISCARD, or TERMINATE. To check for duplicate data, the timestamp information for incoming data is compared to the timestamp information for existing data, and the timestamp information for new, non-duplicate data is saved for future data checking.

FORCE
reviews the incoming data and records information on the duplicate data, but still allows (forces) duplicate data to be processed into the active PDB. The data checking report that is printed in the SAS log provides information about the data that you processed. This value (FORCE) can be used to load data into a table that was not used when processing was done earlier, to load data into a table that was accidentally purged or deleted, and so on.

%CPPROCES and %CMPROCES check whether FORCE is requested by either DUPMODE=FORCE (on the %CPPROCES or %CMPROCES call) or FORCE=YES on the %CPDUPCHK call or both. If either or both are true, the FORCE option in processing is in effect.

Remember to change the DUPMODE= and/or FORCE= settings to their earlier values after you finish using them to allow "duplicate" data into the PDB.

DISCARD

reviews the incoming data and rejects (discards) duplicate data, but allows processing to continue. The data checking report that is printed in the SAS log provides information on the data that was processed and/or discarded.

When DUPMODE=DISCARD, only the FORCE= and TERM= parameters on %CPDUPCHK can affect the FORCE and TERM options in processing.

TERMINATE
reviews the incoming data and terminates the macro if duplicate data is encountered.

%CPPROCES and %CMPROCES check whether TERMINATE is requested by either DUPMODE=TERMINATE (on the %CPPROCES or %CMPROCES call) or TERM=YES on the %CPDUPCHK call or both. If either or both are true, the TERMINATE option in processing is in effect.

In interactive mode, if the TERMINATE option in processing is in effect and duplicate data is detected, then

- the process step terminates
- an ERROR message displays at the end of the SAS log
- the PDB is not deactivated
- the SAS IT Resource Management session continues
- the underlying SAS session continues.

In batch mode, if the TERMINATE option in processing is in effect and duplicate data is detected, the job abends.

Before you can enable duplicate-data checking by means of the %CMPROCES or %CPPROCES macro and/or by means of the %CPDUP* macros, you must implement the data checking macros as described in "Duplicate-Data Checking" on page 671 and, near the end of that topic, the case that applies to your collector or data source.

EXITSRC=*exit-source*
specifies the location where exit routines (source programs) for SAS IT Resource Management are stored. The value of this parameter can be the name of a SAS catalog that is specified in the form of *libref.catname*, or it can be a specified location in your operating environment. For UNIX or Windows, this is the complete pathname of a directory. For z/OS, this is the fully qualified name of a partitioned data set.

If you specify a SAS catalog name, then use a SAS LIBNAME statement in order to associate the *libref* with the SAS library that contains the catalog.

If you want to process your data without using exit routines, then do not specify this parameter.

For the %CSPROCES and %CWPROCES macros, observations with duplicate values for the BY list variables are removed by default. If you want to prevent the removal of duplicate observations, use the PROC180 exit point that is described in "Exit Points for %CSPROCES and %CWPROCES" in the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

For more information on exit processing, see the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

Note: For the %CSPROCES and %CWPROCES macros, if you use process exits and you process HP OpenView Performance Agent data, then you must specify LOGFMT=SINGLETAB in order for your data to be processed. For more information, see the LOGFMT= parameter. △

GROUPDIR=*location-of-group-directory*

specifies the location of the group directory that contains the group files that are required by SAS IT Resource Management in order to process data from the SiteScope collector.

Note: If this parameter is not specified, *location-of-group-directory* defaults to this location: <SiteScope data location>\groups\. △

RAWDATA=*filename*

specifies the full path and filename of the raw data file that you want to process (UNIX and Windows) or the fully qualified data set name for a sequential data set or the fully qualified PDS member name (z/OS).

If you specify COLLECTR=GENERIC and TOOLNM=SASDS, then you must specify the GENLIB= parameter instead of the RAWDATA= parameter.

If you specify COLLECTR=SITESCOP, then the filename is the location of the SiteScope log. (The GROUPDIR= parameter specifies the location of the SiteScope group directory, which contains the group files that are necessary for processing the data from the SiteScope collector. If the GROUPDIR= parameter is not specified, then the group files that are in <SiteScope location>\groups\ are used.)

For any other combination of COLLECTR= and TOOLNM= values, do not use the GENLIB= parameter. Instead use the RAWDATA= parameter or define a fileref named RAWDATA that references the location of the raw data. For example, you can use the SAS FILENAME statement in order to create the RAWDATA fileref:

```
FILENAME RAWDATA 'location-of-raw-data';
```

For information on how to download SMF data from z/OS to UNIX or Windows, see either the "Collector Updates" or "SAS IT Resource Management Collector Updates" section of the *SAS IT Resource Management Server Setup Guide*. In that section, follow this path:

Previous Updates to SAS IT Resource Management Collectors - Click Here ▶ IT Service Vision 2.5 Updates ▶ SMF Data Processing on UNIX and Windows

For instructions on processing raw data for other supplied collectors, such as NTSMF and SAPR3, see the collector-specific information in the *SAS IT Resource Management Server Setup Guide* and its Collector Updates section. Also, see the description of the COLLECTR= parameter.

_RC=*macro-var-name*

specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

TOOLNM=*tool-name*

typically specifies the name of the software that will be used to transform the raw data and to stage the data into SAS data sets or data set views. The value that is specified for this parameter is based on the value that you specify for the COLLECTR= parameter.

For more information about values of the TOOLNM= parameter, see the COLLECTR= parameter. The COLLECTR= parameter contains a table of COLLECTR= and TOOLNM= combinations and also contains collector-specific and macro-specific notes.

Note: Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*. △

%CPPROCES Notes

In general, %CPPROCES processes the data into the PDB's detail level, and a call to %CPREDUCE (with DETAIL=DETAIL, by default) then summarizes the data to the day, week, month, and year levels. (In the case of Weblog data, %CPPROCES processes the data into the PDB's COLLECT library, and a call to %CPREDUCE (with DETAIL=COLLECT) then summarizes the data to the day, week, month, and year levels. The detail level is not used for Weblog data.)

The %CPPROCES macro does not clean up data sets that are written to COLLECT when it has completed. This can be useful when reducing directly from COLLECT and bypassing DETAIL. However, retaining the data sets can be a problem because the data sets occupy disk space until they are deleted. The CPSTGEKP macro variable enables you to specify whether the staged data is retained in its staged location or deleted at the end of the CPPROCES macro.

If you want to retain the staged data sets, specify the following before the %CPPROCES macro:

```
%let cpstgekp=Y;
```

This is the default for all %CPPROCES collectors other than SMF.

If you want to delete all data in the staged location, specify the following before the %CPPROCES macro:

```
%let cpstgekp=N;
```

This is the default when the COLLECTR= value is SMF.

Note: For information about setting age limits for your data and information about the reference date for age limits, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels"Overview of Levels" in the *SAS IT Resource Management User's Guide*. △

For more information about setting up to process data from a particular data collector or data source, see the section "Setting Up the Server" in the chapter "Setup: Introduction" in the *SAS IT Resource Management User's Guide*. Then, follow the references from your setup case to the *SAS IT Resource Management Server Setup Guide*. Also, see the description of the COLLECTR= parameter.

For more information about using process exits when processing your data, see the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

For more information about duplicate-data checking during processing, see “Duplicate-Data Checking” on page 671.

The portable process subsystem that is used by %CPPROCES has implemented support in order to prevent future data from being inadvertently processed into a PDB and thus causing existing data to be aged out of the PDB. For information on using the YEARCUTOFF SAS System option and the CPFUTURE macro variable for testing future data, such as end-of-year data, see the section “Using the YEARCUTOFF SAS System Option” in the chapter “Administration: Working with Data” in the *SAS IT Resource Management User’s Guide* and also refer to the CPFUTURE global macro variable in “Global and Local Macro Variables” on page 6.

%CPPROCES Examples

Example 1: Processing Data for a Table

This example processes data for a table that is named *TABABC* into the detail level of the PDB. The data has already been staged into the SAS library that is named WORK.

```
%cpproces(TABABC,genlib=WORK) ;
```

Example 2: Processing with COLLECTR=WEBLOG on UNIX

If you specify COLLECTR=WEBLOG and TOOLNM=CLF, then you must set a macro variable that is named MACHINE before you run %CPPROCES.

```
%let MACHINE=webserv03;
%CPPROCES(WEBCL1, collectr=WEBLOG, toolnm=CLF,
  rawdata=/usr/tmp/weblog.txt);
```

Example 3: Processing with COLLECTR=SAPR3 - Single Raw Data File on UNIX

This example processes one raw data file by using COLLECTR=SAPR3.

```
%let SAPSYSNM=FINANCE;
%let SAPSYSNR=01;
%let SAPVER=3.0c;
%let SAPHOST=sapsrv01;
%CPPROCES(  sapr3s,
            collectr=sapr3,
            toolnm=statbin,
            rawdata=/usr/tmp/statfile  );
```

Example 4: Processing with COLLECTR=SAPR3 - Multiple Raw Data Files on UNIX

This example processes multiple raw data files by using COLLECTR=SAPR3.

```
%let SAPVER=3.0c;
%CPPROCES(  sapr3s,
            collectr=sapr3,
            toolnm=statbin,
            rawdata=/usr/tmp/statfiles/  );
```

Example 5: Processing with COLLECTR=SAPR3 - Multiple Raw Data Files on Windows

This example processes multiple raw data files by using COLLECTR=SAPR3.

```
%let SAPPLAT=WIN;
%let SAPVER=3.0c;
%CPPROCES(  sapr3s,
            collectr=sapr3,
            toolnm=statbin,
            rawdata=e:\sap\tmp\statfiles\ );
```

Example 6: Processing with COLLECTR=SAPR3 - Multiple Raw Data Files on z/OS

This example processes multiple raw data files by using COLLECTR=SAPR3.

```
%let SAPSYSNM=SAPSRV54;
%let SAPSYSNR=01;
%let SAPVER=3.0c;
%let SAPHOST=;
%CPPROCES(  sapr3s,
            collectr=sapr3,
            toolnm=statbin,
            rawdata=ops.sap.ported.sapfiles );   /* this would be a PDS */
```

Example 7: Processing with COLLECTR=NTSMF

This example processes one raw data file by using COLLECTR=NTSMF.

```
%CPPROCES( ,collectr=ntsmf
           ,rawdata=c:\ntsmfdata\machine.9902091041.smf
           ,toolnm=sasds);
```

This example processes all NTSMF files in the folder that is specified in the RAWDATA= parameter.

```
%CPPROCES( ,collectr=ntsmf
           ,rawdata=c:\ntsmfdata\
           ,toolnm=sasds);
```

Example 8: Processing SMF data on UNIX or Windows

This example on Windows processes data from a raw data file (c:\smf\smfdata\07JUL2001.dat) that was downloaded from z/OS. The data is processed into table XTY70 by using COLLECTR=SMF,TOOLNM=MXG.

```
%CPSTART(MXGLIB=c:\mxglib,
         MXGSRC=('c:\mxgsrc'),
         PDB=pdb-smf) ;
FILENAME SMF 'c:\smf\smfdata\07JUL2001.dat'
         RECFM=S370VBS LRECL=32760;
%CPPROCES(XTY70,
          COLLECTR=SMF,
          TOOLNM=MXG) ;
```

Another way to accomplish the same task is shown below:

```
%CPSTART(MXGLIB=c:\mxglib,
         MXGSRC=('c:\mxgsrc'),
```

```
          PDB=pdb-smf) ;
%CPPROCES(XTY70,
           RAWDATA=c:\smf\smfdata\07JUL2001.dat,
           COLLECTR=SMF,
           TOOLNM=MXG) ;
```

This second way automatically assigns the correct RECFM= and LRECL= parameters to the SMF data.

Note: To process SMF data on UNIX and Windows, you must use the %CPPROCES macro. (To process SMF data on z/OS, you can use either the %CPPROCES macro or the %CMPROCES macro.) △

Note: For information on how to download SMF data from z/OS to UNIX or Windows, see either the "Collector Updates" or "SAS IT Resource Management Collector Updates" section of the *SAS IT Resource Management Server Setup Guide*. In that section, follow this path:
Previous Updates to SAS IT Resource Management Collectors - Click Here ▶ IT Service Vision 2.5 Updates ▶ SMF Data Processing on UNIX and Windows △

%CPRAWLST

Produces a list of records from a raw data file

%CPRAWLST Overview

The %CPRAWLST macro provides a list output from your raw data file. This macro must run after the %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES macro because the process macro provides the name of the *rawdata* file. (The %CPRAWLST macro requires that the file whose data you want to dump has the fileref RAWDATA.) The %CPRAWLST macro is useful in debugging or diagnosing problems with raw data. When you run this macro, the first 10 "lines" of raw data are listed by default, or you can specify the number of "lines" to display.

%CPRAWLST Syntax

%CPRAWLST(
    <**NLINES=***number-of-lines*>);

Details

NLINES=*number-of-lines*
is the number of "lines" that you want to list from the raw data file. *The default number of "lines" is 10*.

A "line" is defined as 16 bytes of data. For example, if you have a line that is 320 bytes long, then specify **`nlines=20`** in order to display the whole line.

%CPRAWLST Example

This example processes UNIX accounting data for an HP/UX system and uses a *pipe* command to prefix the machine name to the raw file. If the %CSPROCES macro fails,

then call the %CPRAWLST macro in order to display the output from the PIPE command and determine where the error occurred. Because the NLINES= parameter is not specified, only the first 10 "lines" of the output are displayed.

```
%csproces(pipe 'echo "machine=disk1";cat /usr/pacct',
          acchp7,
          passwd=/nfs/disk1/etc/passwd, group=/nfs/disk1/etc/group,
          collectr=accton,
          _rc=retcode);
%put csproces return code= &retcode;

%cprawlst;
```

%CPREDUCE

Reduces data from the detail level to one or more summary levels

%CPREDUCE Overview

The %CPREDUCE macro summarizes data in the detail level of the PDB into the day, week, month, and year levels of the PDB. This summary takes place for the specified tables or, if none are specified, then for all tables that meet the following criteria:

- The table contains data that has been processed into the detail level since the last reduction.
- The table's KEPT= parameter is set to YES.
- The value of the AGELIMIT= parameter is greater than zero in at least one of the day, week, month, or year levels.
- The table has at least one variable that has a KEPT status of YES.

The %CPREDUCE macro also removes data and statistics that have exceeded the age limit for the day, week, month, and year levels. As it finishes, %CPREDUCE removes data from the table's detail level if and only if the detail-level age limit of the table is zero.

%CPREDUCE Syntax

%CPREDUCE(
<*tablist*>
<,**CKPTINIT=**NO | YES>
<,**DETAIL=***libref*>
<,**_RC=***macro-var-name*>);

Details

tablist
specifies which tables to reduce. Separate the table names with blanks. *This is not a required positional parameter.*

If you omit the list, then the macro reduces the detail data in all tables in your PDB that meet the following criteria. If you specify a list of tables, then the detail data in the specified tables will be reduced if they meet the following criteria:

- The table contains data that has been processed into the detail level since the last reduction.
- The KEPT= parameter is set to YES.
- The value of the AGELIMIT= parameter is greater than zero in at least one of the day, week, month, or year levels.
- The table has at least one variable that has a KEPT status of YES and has at least one statistic that is selected.

CKPTINIT=NO | YES
indicates whether to delete all checkpoint records. When you run %CPREDUCE, flags are placed in the checkpoint data set (DICTLIB.CKPTINFO) in order to indicate the completion status of reduction steps. *CKPTINIT=NO is the default*.

If a %CPREDUCE failure occurs, then the checkpoint data set contains information on the failure so that the next %CPREDUCE can automatically recover from the failure. Intermediate data sets that are used for reduction also remain in the SAS data library PDBWORK, so that data is not lost when the automatic %CPREDUCE recovery procedure takes place.

CKPTINIT=NO means keep the checkpoint file intact and do not delete the data sets in PDBWORK.

CKPTINIT=YES means clear the checkpoint file and delete the data sets in PDBWORK.

Additional information on using this macro in recovery from an error condition is available in "Troubleshooting Batch Jobs That Fail" on page 655.

CAUTION:
If you use the CKPTINIT=YES option, then you might lose some data for the table and time period that are being reduced. You will not receive a warning that the data is being deleted. Do not use this option unless you can accept losing some summary data for the table and time period that are being reduced when the failure takes place, or unless you are instructed to do so by SAS Technical Support. △

If AGELIMIT=0 at the detail level, then your data is stored in the detail level until the next time that the %CPREDUCE macro or %CxPROCES macro runs. Therefore, if %CPREDUCE runs immediately after the %CxPROCES macro runs, then the data is processed, reduced into the summary levels, and removed from the detail level. However, if the %CxPROCES macro runs again before the data has been reduced (by means of %CPREDUCE), then the existing data in detail is deleted before it can be reduced into the summary levels. This means that if %CPREDUCE fails for one or more tables that have an AGELIMIT=0 at the detail level, then you should fix the cause of the failure and rerun %CPREDUCE before the next process step (%CxPROCES) runs. Using AGELIMIT=0 at the detail level is useful if you are collecting and processing large amounts of data on a daily basis, but in order to avoid losing data at the summary levels, always reduce the data immediately after it is processed.

Note:

1. %CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES.
2. For information on setting age limits for your data, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*.

△

DETAIL=*libref*

specifies the libref of the existing SAS library that has the detail-level data for the tables that you want to reduce. The libref must be defined by the time that this macro is called.

Note: If the value of this parameter is not DETAIL, then it is referring to *external detail*. △

The default value is DETAIL.

Specifying a libref other than DETAIL is a method for bypassing the %CxPROCES step. This method is used if you want to load data into the summary (reduction) levels directly from existing SAS data sets or views, and do not need to store detail-level data in the PDB.

Note: %CxPROCES refers to %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES. △

In the library that is indicated by this libref, the %CPREDUCE macro looks for the SAS data sets or views that are named in the form of **`tablename`**D (that is, the name of the table with the letter D appended). This is the same data set name that is normally found in the DETAIL library. The format of the data sets or views in the specified DETAIL= library should be the same as the format of the same-named data sets in the PDB's DETAIL library, including the variable names.

If you want to reduce data in tables that were created by the %CPAVAIL macro, specify DETAIL=ADMIN for those tables. Similarly, if you want to reduce weblog data, specify DETAIL=COLLECT for those tables. And for data to be reduced after calling the %CMEXTDET parameter, set the DETAIL= parameter to the value that is specified by the VIEWLIB= parameter in the call to %CMEXTDET.

Each time that you call the %CPREDUCE macro, you can use the DETAIL= parameter to obtain data from a different library. Thus, if you want to reduce data from more than one library, you must call the %CPREDUCE macro more than one time. Call the %CPREDUCE macro one time for each set of tables that have a different libref for the SAS library that contains the detail-level data.

_RC=*macro-var-name*

specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPREDUCE Notes

Remember to back up the PDB after you process and reduce the data.

If you want all your data to be reduced, then you must run the reduce step at least as often as the shortest detail-level age limit of your tables. For example, if your shortest detail-level age limit is zero, then you must run the reduce task before the next process task runs. Otherwise, the process step will age out the data before the reduce step has an opportunity to see it.

For information about setting age limits for your data and information about the reference date for age limits, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*.

Note: For more information about external detail, see "%CMEXTDET" on page 31. △

%CPREDUCE Examples

Example 1

This example reduces the data in the detail level of all tables that meet the reduction criteria to the summary levels that are specified in the data dictionary for the PDB.

```
/*reduce "all" tables in the active PDB */
%cpreduce();
```

Example 2

This example reduces data in tables TABLE3 and TABLE8 and keeps the checkpoint data set. If an error occurred in a previous reduction task, then %CPREDUCE starts where it stopped; if the error has been fixed, then the macro continues.

```
/*reduce two tables in the active PDB */
%cpreduce(table3 table8);
```

Example 3

Use this method only if this is a practice PDB, if you can accept possible data loss, or if you are instructed to do so by SAS Technical Support.

This example turns off the error flag in the checkpoint data set, deletes the data sets in the PDBWORK library, and reduces the data in all the tables in the PDB. Some data may be lost from the day, week, month, or year levels. The data that may be lost consists of updates to statistics that were being calculated for a level of the table when the error occurred.

```
 /*may lose data from a previous
interrupted reduction job */
%cpreduce(,ckptinit=yes);
```

%CPRENAME

Renames tables or variables in the active PDB

%CPRENAME Overview

The %CPRENAME macro renames tables within the active PDB or renames variables within a table in the active PDB, depending on which parameters you specify with this macro. %CPRENAME also renames the statistics variables at the summary levels.

When you use %CPRENAME to rename one or more tables, the OLDTABL= and NEWTABL= parameters are required and the BLDVIEWS= and _RC= parameters are optional.

When you use %CPRENAME= to rename one or more variables, the TABLE=, OLDVARL=, and NEWVARL= parameters are required and the BLDVIEWS= and _RC= parameters are optional.

%CPRENAME Syntax

%CPRENAME(
NEWTABL=*new-table-list*
,**NEWVARL=***new-var-list*
,**OLDTABL=***old-table-list*
,**OLDVARL=***old-var-list*
,**TABLE=***table-name*
<,**BLDVIEWS=**Y | N>
<,**_RC=***macro-var-name*>);

Details

NEWTABL=*new-tab-list*
specifies a list of new table names. The table names must be separated by blanks and must not already exist in the active PDB. The number of tables that are specified in this list must match the number of tables that are specified for the OLDTABL= parameter. The table names that are listed in the OLDTABL= list are changed to match the corresponding positional table names in the NEWTABL= list.

The name must begin with a letter (U is recommended for user-defined tables) and can contain letters and numbers. The maximum length is seven characters. The name is not case sensitive.

NEWVARL=*new-var-list*
specifies a list of new variable names for the corresponding entries that are specified in the OLDVARL= parameter. The variable names must not already exist in the active PDB in the table that is specified by the TABLE= parameter. Use a blank space to separate the names in the list.

The lists that are specified in OLDVARL= and NEWVARL= must have the same number of entries. The variable names in the OLDVARL= parameter list are changed to match the corresponding positional names in the NEWVARL= list.

The name must begin with a letter and can contain letters and numbers. The maximum length is seven characters. The name is not case sensitive.

OLDTABL=*old-table-list*
lists the names of the tables that you want to rename. The table names must be separated by one or more blanks and must already exist in the active PDB. The number of tables that are specified in this list must match the number of tables that are specified for the NEWTABL= parameter. The table names in the OLDTABL= parameter list are changed to match the corresponding positional names in the NEWTABL= list.

OLDVARL=*old-var-list*
lists the names of the variables that you want to rename. The variable names must already exist in the active PDB in the table that is specified by the TABLE= parameter. Separate the names in the list with one or more blank spaces.

The number of variables that are specified in this list must match the number of variables that are specified for the NEWVARL= parameter. The variable names

in the OLDVARL= parameter list are changed to match the corresponding positional names in the NEWVARL= list.

TABLE=*table-name*
specifies the name of the table that contains the variables that you want to rename. The table must already exist in the active PDB.

BLDVIEWS=Y | N
specifies whether or not to allow the views to be rebuilt automatically when all your specified changes have been made. *The default is N, which means that the views will not automatically be rebuilt.*

If you specify BLDVIEWS=N, then you should ensure that the views are rebuilt. To rebuild the views yourself, run %CPDDUTL and the BUILD VIEWS control statement against the active PDB. The next time that the process step or reduce step runs, it will rebuild the views if you haven't rebuilt them yourself. Do not report on data until the views are rebuilt.

If you specify BLDVIEWS=Y, then the views are rebuilt unless the OLDVARL= parameter is specified and one or more of the variables in its list is a formula variable or derived variable. In this case, everything has been renamed in the data dictionary except in the source statements for the formula variables and derived variables. To replace the source statements, you can use the GUI, or you can use %CPDDUTL to apply UPDATE FORMULA and UPDATE DERIVED control statements for each formula and derived variable that you rename. In these control statements, use the SOURCE= parameter to define the new name of the variable.

For example, use

```
new-variable-name=...;
```

instead of

```
old-variable-name=...;
```

You do not need to follow these control statements with a BUILD VIEWS control statement, because the updates automatically cause the views to be rebuilt.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPRENAME Notes

Always make a backup of your PDB before you run this macro.

If any of the variables that are being renamed is a formula or derived variable, then the views will not be rebuilt even if BLDVIEWS=Y. Use the %CPDDUTL control statements UPDATE FORMULA and or UPDATE DERIVED to edit the source for the formula or derived variable(s), after which %CPDDUTL will automatically rebuild the views.

Do not use this macro if you are having any difficulties with your PDB, such as an error during reduction.

Do not use this macro between runs of process and reduction.

This macro will not rename variables in report definitions or rule definitions.

CAUTION:

This macro will verify that your proposed new names do not already exist, but it cannot detect cases of circular renaming, or renaming that is dependent upon earlier renaming in the same %CPRENAME invocation. Therefore, within one call to this macro, avoid circular renaming. Try to keep your changes simple. △

%CPRENAME Examples

Rename table XTY70 to UTY70:

```
%cprename(oldtabl=XTY70,newtabl=UTY70,_RC=retcode);
```

Rename table XTY70 to UTY70 and table XTY30 to UTY30:

```
%cprename(oldtabl=XTY70 XTY30,newtabl=UTY70 UTY30);
```

Rename variable ABC to XYZ, in table XTY70:

```
%cprename(table=XTY70,oldvarl=ABC,newvarl=XYZ);
```

Rename variable ABC to XYZ, and variable ZZZ to variable AAA, both in table XTY70:

```
%cprename(table=XTY70,oldvarl=ABC ZZZ,newvarl=XYZ AAA);
```

%CPRPT2DD

Creates a table list and data dictionary control statements based on an HP OpenView Performance Agent (HP-OVPA) report parameter file (UNIX and Windows only)

%CPRPT2DD Overview

The %CPRPT2DD macro generates information that you can use to tailor your PDB for the data that you want to use from HP OpenView Performance Agent (HPOVPA).

The data that you want to use is indicated in the HPOVPA report parameter file. Given that file, the %CPRPT2DD macro

- creates a list of tables.

  These are the SAS IT Resource Management tables that contain one or more variables that correspond to the variables described in the report parameter file.
- creates blocks of %CPDDUTL control statements, one block for each table in the list.

Each block's first control statement is a SET TABLE control statement that identifies the table.

The remaining control statements in a block are UPDATE VARIABLE or DELETE VARIABLE control statements that apply to variables that are associated with the table.

Note: The %CPDDUTL control statements that are generated will not affect FORMULA or DERIVED variables; therefore, variables which are used by DERIVED or FORMULA variables should be set with the same disposition as those DERIVED or FORMULA variables. △

- For each variable that *does* correspond to a variable described in the report parameter file, the %CPRPT2DD macro generates an UPDATE VARIABLE control statement that sets **`KEPT=YES`**, which indicates that this variable *is* to be used.
- For each table variable that *does not* correspond to a variable described in the report parameter file, the %CPRPT2DD macro generates either an UPDATE VARIABLE control statement or a DELETE VARIABLE control statement.
  - If the DELVARS= parameter is set to NO (or is not specified), the %CPRPT2DD macro generates an UPDATE VARIABLE control statement that sets **`KEPT=NO`**, which indicates that this variable is not to be used. (Later, if you want to use this variable, you can change its kept status to Yes.) *This is the recommended value.*
  - If the DELVARS= parameter is set to YES, the %CPRPT2DD macro generates a DELETE VARIABLE control statement, which indicates that this variable is to be removed from its table. (Later, if you want to use this variable, you must re-create the variable in the table and then set its kept status to **`Yes`**.)

Then, you can use the table list to add the appropriate supplied table definitions (and their associated supplied variable definitions) to a PDB. And you can use the control statements to specify the appropriate use (and non-use) of the associated variable definitions.

%CPRPT2DD Syntax

%CPRPT2DD(
OUTDDUTL=*file-to-contain-%CPDDUTL-statements*
,**REPTFILE=***location-of-HPOVPA-report-parm-file*
<,**DELVARS=**NO | YES>
<,**_RC=***macro-var-name*>
<,**_TABLIST=***macro-var-name*>);

Details

OUTDDUTL=
file-to-contain-%CPDDUTL-statements
specifies the complete filename and pathname of the file that contains %CPDDUTL control statements. The %CPRPT2DD macro writes this file.
This parameter is required.

REPTFILE=

location-of-HPOVPA-report-parm-file
specifies the complete filename and pathname of the HP OpenView Performance Agent report parameter file that you are using with your HP OpenView Performance Agent extract program. The %CPRPT2DD macro reads this file.
This parameter is required.

DELVARS=*YES | NO*
determines what should happen to a SAS IT Resource Management variable that does not correspond to the HPOVPA report parameter file.
If DELVARS=NO, the kept status of the variable is to be set to NO, but the variable's definition continues to be associated with its table's definition.
If DELVARS=YES, the variable is deleted from the table.
This parameter is optional. The default value is NO.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △
If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.
For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.
You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).
There is no default value for the name of the macro variable.

_TABLIST=*macro-var-name*
specifies the name of a macro variable that is to contain the list of SAS IT Resource Management supplied tables whose definitions are associated with variables that correspond to variables in the HP OpenView Performance Agent report parameter file. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).
If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

%CPRPT2DD Notes

The HP OpenView Performance Agent report parameter file lists, for each data type, all the metrics that are going to be extracted with the extract command.
Typically, to use this macro you would follow these steps:

1. Call the %CPSTART macro to allocate a new PDB (with no table definitions).
2. Call the %CPRPT2DD macro to generate the list of tables and the file of control statements.

3 Again, call the %CPSTART macro to reactivate the new PDB (because the call to %CPRPT2DD deactivates the PDB).
4 Call the %CSPROCES or %CWPROCES macro with no input data but with the table list. The %CSPROCES or %CWPROCES macro will first add the supplied table definitions from the master data dictionary. Then, the %CSPROCES or %CWPROCES macro will terminate because the input data file is empty.
5 Call the %CPDDUTL macro with the file of control statements. The %CPDDUTL macro will apply the control statements to the supplied tables and tailor them to correspond to the report parameters file.

The result is a PDB that is ready for production use.

%CPRPT2DD Example

```
*...;
* Edit the values below;
*...;

* Set PDBLOC to the pathname of your PDB;
%let PDBLOC=/your/pdb;

* Set REPTMWA to the name of the HPOVPA report parm file to read;
%let REPTMWA=/your/reptfile.mwa;

*...;
* End of user editable values;
*...;

*...;
* Advertise;
*...;
%put NOTE: PDBLOC is &PDBLOC;
%put NOTE: REPTMWA is &REPTMWA;

*...;
* Establish SAS IT Resource Management environment;
*...;
%CPSTART(pdb=&PDBLOC, mode=batch);


*...;
* Establish values for UNIX vs Windows operating systems ;
*...;
%macro UNIXPC;
   %global slash deltree tempdir cxproces;
   %if %substr(&SYSSCP,1,3) = WIN | %substr(&SYSSCP,1,3) = OS2  %then %do;
      %let slash = \;
      %let deltree  = deltree / y;
      options noxwait;
      %let TEMPDIR=%SYSGET(TEMP);
      %if %BQUOTE(&TEMPDIR)= %then %do;
          %let TEMPDIR=%STR( c:\doc);
          %put NOTE: TEMP environment variable not set. Using &TEMPDIR..;
      %end;
```

```
      %let CXPROCES=CWPROCES;
   %end;
   %else %do;
      %let slash = /;
      %let deltree = rm  ---r;
      %let TEMPDIR=%BQUOTE(/tmp);
      %let CXPROCES=CSPROCES;
   %end;
%mend UNIXPC;
%UNIXPC;

*...;
* Create %CPDDUTL control statements from the report parm file;
*...;
%let TABLIST=;

%CPRPT2DD( reptfile=&REPTMWA,
           outddutl=&TEMPDIR&SLASH.ddutl.mwa,
           _tablist=&TABLIST.,
           _rc=RETC ) ;
%CPFAILIF ( &RETC ne 0 );

*...;
* Allocate the PDB again (CPRPT2DD de-allocates it);
*...;
%CPSTART( pdb=&PDBLOC,
          access=write,
          mode=batch,
          _rc=RETC );
%CPFAILIF( &RETC ne 0 );

*...;
* Add table definitions not already in PDB;
*...;
%&CXPROCES( ,&TABLIST, collectr=HP-OVPA, _RC=RETC );
%CPFAILIF( &RETC ne 0 );

*...;
* Run the generated control statements to keep or not keep the metrics;
*...;
%CPDDUTL( filename=%QUOTE(&TEMPDIR&slash.ddutl.mwa), list=yes );

*...;
* Clean up;
*...;

%CPXEQCMD( &deltree &TEMPDIR&SLASH.ddutl.mwa );
```

%CPRPTPKG

Reports on the contents of a SAS IT Resource Management package

%CPRPTPKG Overview

The %CPRPTPKG macro reads the contents of a SAS IT Resource Management package. For example, if the %CPPKGCOL macro created the package, the package contains user-written collector-support entities for a particular data collector or data source.

Then, the %CPRPTPKG writes a report on the contents of the package. The report is written in HTML. The report includes the contents of any documentation that is in the package. For example, if the %CPPKGCOL macro created the package, then the report lists the collector name and tool name for which the package was created, lists which collector-support entities are in the package, and also contains the contents of the documentation entity in the package, if there is a documentation entity in the package.

%CPRPTPKG Syntax

%CPRPTPKG(
DOCLOC=*location-of-report*
,PACKFILE=*fileref*
<,**_RC=**>);

Details

DOCLOC=*location-of-report*
specifies the location where the report is to be written.

Note: Do not use quotation marks around the location. △
This parameter is required.
The form for specifying the location is operating system dependent:

- On UNIX and Windows, specify the full path and name of a directory. The directory must already exist.
- On z/OS, specify the fully qualified name of a PDS or specify the full path and name of a directory in the z/OS UNIX File System area. The PDS or directory must already exist.

PACKFILE=*fileref*
specifies a fileref. The fileref points to the external file where the package will be stored (if the package is being created) or is stored (if the package is being reported on or installed). The fileref must already exist by the time that the macro is called.

For example, suppose that you are creating in Windows a package of collector-supported entities and you want to use a fileref of **pkgfile**. Before your call to the %CPPKGCOL macro, you must have a SAS FILENAME statement similar to this one:

```
FILENAME pkgfile 'c:\ucollpkg.xpt' ;
```

where **c:\ucollpkg.xpt** is the location where the package is to be written.

Similarly, suppose that you are reporting on the same package, which has been FTP'd in binary mode to UNIX, and you want to use a fileref of **unixpack**. Before your call to the %CPRPTPKG macro, you must have a SAS FILENAME statement similar to this one:

```
FILENAME unixpack '\usr\itsv\ucoll\ucollpkg.xpt' ;
```

where **\usr\itsv\ucoll\ucollpkg.xpt** is the location of the package.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPRPTPKG Notes

The report generated by %CPRPTPKG is typically used as temporary documentation, and deleted eventually. The report generated by %CPINSPKG is typically used as permanent documentation. Both reports are identical.

For more information about collectors and collector support, see the section "Setting Up the Server" in the chapter "Setup: Introduction" in the *SAS IT Resource Management User's Guide*. For more information about packaging and installing collector support, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*, "%CPINSPKG" on page 124, and "%CPPKGCOL" on page 144.

%CPRPTPKG Example

Suppose that the package location is **`c:\mycol\mypkg.xpt`**, the fileref for that package is to be **`package`**, and the report is to be written to the directory **`c:\mycol\doc`**.

The code should look something like this:

```
FILENAME package 'c:\mycol\mypkg.xpt' ;

%CPRPTPKG(PACKFILE=package
        , DOCLOC=c:\mycol\doc
        , _RC=rptrc ) ;

%PUT CPRPTPKG return code is &rptrc ;
```

%CPSEP

Verifies and, if necessary, adds an operating system–dependent separator to a UNIX or Windows pathname or to a z/OS data set name

%CPSEP Overview

The %CPSEP macro ensures that a user-supplied pathname or user-supplied z/OS data set name has a separator at the end of it so that it can be used as a prefix for building groups of directories or PDSs whose names share a common prefix. If necessary, the macro adds an operating environment–dependent separator to a UNIX or Windows pathname or to a z/OS data set name.

%CPSEP Syntax

%CPSEP(
name);

Details

name
specifies the operating environment name to use as the prefix for the pathname (UNIX or Windows) or for the data set name (z/OS).

%CPSEP Notes

The output of the %CPSEP macro is the adjusted name—that is, the name with exactly one separator at the end. SAS IT Resource Management selects the operating environment–dependent separator as follows:

UNIX - one forward slash (/)

Windows - one backward slash (\)

z/OS - one dot (.)

If the pathname already has one or more separators at its right-most end, then it/they will be replaced by exactly one separator. If there is no separator at the end of the name, then one will be added.

%CPSEP Examples

```
%CPRUNRPT( ..., htmldir=%CPSEP(&CPPDB)jpserv);
```

In the preceding example, if CPPDB is **/my/pdb**, then the html directory will have the following value:

```
/my/pdb/jpserv
```

If CPPDB is **SASGXE.MYPDB.**, then the html directory will have the following value:

```
SASGXE.MYPDB.jpserv
```

If CPPDB is **E:\pdb**, then the html directory will have the following value:

```
E:\pdb\jpserv
```

%CPSTART

Starts SAS IT Resource Management software

%CPSTART Overview

The %CPSTART macro allocates PDB libraries, establishes system options and global macro values for the SAS IT Resource Management application, specifies the active PDB, and optionally starts the SAS IT Resource Management GUI. Thus, the %CPSTART macro establishes your environment before you perform any other tasks within the application.

To start SAS IT Resource Management in batch, you must run the %CPSTART macro. You can submit this macro from the SAS Program Editor window in the SAS window interface, or you can submit from the SAS Program Editor window a command that invokes this application by submitting the %CPSTART macro. That command is **`itrm`**.

If you execute the %CPSTART macro without specifying the PDB= parameter, then the macro uses the PDB that you last accessed. The only exception to this is for z/OS batch jobs, for which there is no default PDB.

The active PDB is the PDB that you have most recently selected to use or act upon and the PDB on which most of the subsequent operations are performed. For example, all the reporting macros report on data that is contained in the active PDB, but the %CPDBCOPY macro cannot use the active PDB as the target PDB.

You can set a different PDB to be the active PDB at any time by using the %CPSTART macro.

%CPSTART Syntax

%CPSTART(
<**ACCESS=**READONLY | WRITE>
<,**DISP=**OLD | SHR>
<,**MODE=**MENU | BATCH>
<,**MXGLIB=***SAS-library*>
<,**MXGSRC=**(*'mxg-userid-sourclib' 'mxg-mxg-sourclib'*)>
<,**PDB=***pdb-name*>
<,**_RC=***macro-var-name*>
<,**REMPROF=***profile-name*>
<,**ROOT=***root-location*>
<,**SHARE=***pdb-server*>
<,**SITEACC=***sitelib-access*>
<,**SITELIB=***site-library*>
<,**SITESHR=***sitelib-server*>);

Details

ACCESS=READONLY | WRITE
specifies the level of access to the performance database. You must specify ACCESS=WRITE if you are processing or reducing data. This parameter is used only in UNIX and Windows operating environments.

To create a new PDB in batch, you can use %CPSTART with the PDB= parameter and ACCESS=WRITE. This will create the root PDB directory and the appropriate subdirectories.

If you are accessing an existing PDB and you do not specify the ACCESS= parameter, then your default level of access is the same level that you had for this PDB the last time you accessed it. If you are accessing an existing PDB that you have not used before, then your default level of access is READONLY.

If you are accessing a remote PDB, then do not specify the ACCESS= parameter.

DISP=OLD | SHR
is the standard IBM disposition value. It applies to the access level of the PDB. This parameter is used only on z/OS.

The default setting is DISP=SHR, which indicates that you and others have read access to the PDB so that multiple users can read the PDB at the same time.

If you specify DISP=OLD, then you have write access to the PDB and therefore you have exclusive control.

MODE=MENU | BATCH
specifies which interface to use when you start SAS IT Resource Management. This value determines how missing values for the location of ROOT= and PDB= are handled, as described below. *The default value is MODE=MENU.*

MODE=MENU
must not be specified when %CPSTART is submitted in batch mode (JCL, shell scripts, etc.), but can be specified when %CPSTART is submitted through the SAS Program Editor window. MODE=MENU invokes the software and starts the SAS IT Resource Management GUI.

Note: On z/OS, if the location that specified in the ROOT= parameter is not found, then you are prompted for the location. For UNIX and Windows environments, the macro terminates. △

Note: For UNIX, Windows, and z/OS environments, if the location of the active PDB (as specified in the PDB= parameter) is not found, then you are prompted for the location. △

MODE=BATCH
is required when %CPSTART is submitted in batch mode (JCL, shell scripts, etc.), and can be specified when %CPSTART is submitted through the SAS Program Editor window. MODE=BATCH invokes the software, but it does not start the SAS IT Resource Management GUI. If the location that is specified in the ROOT= parameter is not found, then this mode terminates the %CPSTART macro.

Note: For UNIX or Windows environments, if the PDB cannot be found and ACCESS=WRITE, then this mode creates an empty PDB; if ACCESS=READONLY, then the macro fails. △

Note: On z/OS, if the PDB cannot be found and DISP=SHR or DISP=OLD, then the macro fails and you receive an error message. For more information, see the DISP= parameter. △

MXGLIB=*SAS-library*
specifies the physical location and name of the SAS library that contains the MXG formats to use with MXG variables. The MXG documentation refers to this library as MXG.FORMATS.

This parameter is required if this SAS IT Resource Management session will process data by using MXG-based staging code and/or report on data by using MXG formats. For more information about MXG, see the %CPSTART Notes section.

MXGSRC=*('mxg-userid-sourclib' 'mxg-mxg-sourclib')*

specifies the physical location and name of one or more MXG source libraries. These are the source libraries that MXG documentation refers to as MXG.USERID.SOURCLIB and MXG.MXG.SOURCLIB.

- On z/OS, these source libraries are PDSs and contain members.
- On UNIX or Windows, these source libraries are directories and contain files.

This parameter is required if this SAS IT Resource Management session will process data by using MXG-based staging code. For more information about MXG, see the %CPSTART Notes section.

In the value of the MXGSRC= parameter, list the source libraries in search order. If the same member name exists in more than one of the source libraries, then the first member that is encountered is the one that is used.

MXG.MXG.SOURCLIB is read-only. Any additions or updates to MXG.MXG.SOURCLIB are placed in MXG.USERID.SOURCLIB. If a site does not have any additions or updates to MXG.MXG.SOURCLIB, the MXG.USERID.SOURCLIB is empty or does not exist, and can be omitted from the list of MXG source libraries. If a site does have additions or updates to MXG.MXG.SOURCLIB, then MXG.USERID.SOURCLIB should be listed in front of MXG.MXG.SOURCLIB.

PDB=*pdb-name*

specifies the name of the active PDB. This includes the complete directory path or high-level qualifier and the PDB name, but it does not include the lowest levels such as ADMIN, DAY, or any of the other libraries within the PDB. *If you omit this parameter on the first call to %CPSTART, then the macro uses the PDB that you last accessed. If you omit this parameter on a second (or later) call to the %CPSTART macro within the same SAS session, then the macro uses the most recently specified PDB.* Refer to the MODE= parameter for more details.

You can switch the active PDB and/or your access level to that PDB in the middle of a batch job by calling the %CPSTART macro again with new values for the PDB= parameter, the ACCESS= or DISP= parameter, and, if necessary, the REMPROF= parameter. If a call to the %CPDEACT macro has deactivated the active PDB (or the call to the %CPSTART macro was unable to activate the specified PDB), then you can activate a different PDB (or the same PDB) by calling %CPSTART again in the same job.

If you start SAS IT Resource Management on a client host and you access a PDB on a remote host, then you must also use the REMPROF= parameter in order to identify the remote profile for the active PDB. You do not need to specify DISP= or ACCESS= parameter. DISP=SHR or ACCESS=READONLY is automatically specified for remote PDBs.

For more information about the steps required to create a PDB, see the section "Setting Up the Server" in the chapter "Setup: Introduction" in the *SAS IT Resource Management User's Guide* and also follow its reference to the chapter that describes the setup case that is appropriate for your data collector or data source.

For z/OS:

To create a new PDB in batch, the nine PDB libraries should be allocated (typically by IEFBR14) before you execute %CPSTART. You may then initialize the PDB by using %CPSTART with the appropriate PDB= parameter and DISP=OLD.

For UNIX and Windows environments:

To create a new PDB in batch mode, use %CPSTART with the appropriate PDB= parameter and ACCESS=WRITE. This will create the PDB and, if necessary, will initialize it.

Although performance databases must be protected from concurrent update access by multiple users (by using careful coding practices or by adding restrictive ACLs), you can provide multi-read/single-write access to a PDB by using the *-filelocks none* option on the SAS command.

If you start SAS with *-filelocks none*, then you can do ad hoc reporting on the PDB at the same time that you process and reduce data. You will sometimes get "data set not found" messages or incomplete graphs (if the date range in the data dictionary does not match the date range in the data itself because an update is in progress). In that case, wait briefly and request the report again.

Also, if you start SAS with *-filelocks fail*, then you can do ad hoc reporting on the PDB at the same time that you process and reduce data. You will sometimes get "lock fail" messages. In that case, wait briefly and request the report again. In either case, you are not required to run SAS/SHARE.

Note: The value of none for the *-filelocks* option allows concurrent updates from multiple SAS sessions, and this could cause files to become corrupt. Specifying *-filelocks fail* also DOES NOT protect against corruption of a PDB, because a PDB consists of multiple SAS data sets (multiple files) and the data sets can get out of sync if multiple SAS sessions attempt to update them concurrently. To prevent corruption, you *must* ensure that only one process or reduce task at a time attempts to update a PDB. Thus, if you have more than one process-and-reduce batch job that runs against the PDB, then schedule the jobs at separate times. △

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

REMPROF=*profile-name*
specifies the name of the remote server profile to use with the PDB that specified in the PDB= parameter. *You must specify this parameter each time that you want to activate a remote PDB.* If you do not specify this parameter, the PDB that you specify with the PDB= parameter is assumed to be local. Thus, you do not need to specify this parameter if you are accessing a local PDB. A local PDB is a PDB that resides on a disk that is attached to your current host or a PDB that is available through a service such as Network Neighborhood, NFS, AFS, or DFS.

This parameter is supported only in environments where a SAS IT Resource Management client license is installed.

The profile name can be a maximum of eight characters long and must be an existing profile. The profile is a collection of information about the server host on which the specified PDB is located and is used when you access that remote PDB. The profile identifies things such as the location of the server, the communication method used to access the server, the server host, and more.

To create or modify a remote server profile, select the Administration tab in the SAS IT Resource Management GUI and then select the Manage PDBs task on that tab. From the Manage PDBs window, select a PDB for which the remote server profile is appropriate. Then select **Locals** from the pull-down menus, and select **Remote Profile**. From the Remote Server Connection Profiles window, you can create and modify a server profile or associate a profile with a PDB.

If you specify this parameter, then the options that you specified when you created the remote profile will override some of the parameters on the %CPSTART macro. For example, if you have specified a site library or an MXG library for this remote profile, then those values are used instead of the values that you specify for the MXGLIB= and SITELIB= parameters on %CPSTART. Additionally, if you have specified a remote site library within the remote profile, then the SITEACC= and SITESHR= parameters on %CPSTART are ignored, because remote access is read-only.

If you decide not only to access data from a PDB that is on a remote host but also to do other work (such as run report definitions) on that remote host, you can submit the remote work by wrapping it with *rsubmit;* and *endrsubmit;*. For example, if you want to run a call to %CPPLOT1 on the same host as the PDB from which it obtains its data, you could add the following code to your batch job:

```
rsubmit;
   %CPPLOT1(...);
endrsubmit;
```

Note: Do not remote submit a call to a report macro if both of these conditions are true: the call uses OUTMODE=WEB, and the operating environment on the remote host is z/OS. △

Note: If a call to %CPRUNRPT has its RSUBMIT= parameter set to YES, do not wrap the call with *rsubmit;* and *endrsubmit;*. %CPRUNRPT will wrap itself if RSUBMIT=YES. △

ROOT=*root-location*
is the physical location where the SAS IT Resource Management software is installed on your system. This includes the complete directory path or high-level qualifiers where PGMLIB is installed, but not the name PGMLIB.

On z/OS, this parameter is required. You must use the ROOT= parameter to specify the location of SAS IT Resource Management at your site.

For UNIX or Windows environments, this parameter is required if you install SAS IT Resource Management in a location other than the default installation location. If you install SAS IT Resource Management in the default installation location, then you do not need to specify this parameter.

For more information, see the installation instructions that accompany your software or ask the SAS Installation Representative at your site.

SHARE=*pdb-server*
names the SAS/SHARE server that is to be used to activate the PDB specified by the PDB= parameter. SAS/SHARE software enables multiple users to access SAS data libraries concurrently. Use this parameter only if you want to use SAS/SHARE software with a SAS IT Resource Management PDB at your site.

Caution should be exercised when you use SAS/SHARE in the SAS IT Resource Management environment, especially during periods when SAS data is being updated by batch processes.

This parameter is used only if you are running SAS IT Resource Management server software on z/OS.

Note: If you are experiencing performance problems with %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES and/or with %CPREDUCE, consider shutting down the SAS/SHARE server in order to access the PDB directly while process and reduce are running. For information, see the documentation for SAS/SHARE. △

For more information, check with your SAS Installation Representative or with SAS Technical Support.

SITEACC=*sitelib-access*
specifies the level of access to the site library.

Valid options for UNIX and Windows environments are READONLY or WRITE. For z/OS, valid options are SHR and OLD. Do not place quotation marks around the value of this parameter.

SHR (z/OS) or READONLY (UNIX and Windows) specifies that you and others have concurrent read access to the SITELIB library at your site. OLD (z/OS) or WRITE (UNIX and Windows) specifies that you have exclusive control and write access to the SITELIB library at your site.

If you specify the SITELIB= parameter but do not specify the SITEACC= parameter, then the system provides WRITE (UNIX and Windows) or OLD (z/OS) access to the specified SITELIB library. If you do not specify the SITELIB= or the SITEACC= parameters, then the system provides READONLY (UNIX or Windows) or SHR (z/OS) access to the default SITELIB library.

SITELIB=*site-library*
is the physical location and name of your SITELIB library. This includes the complete directory path or high-level qualifiers of the site library and the library name. The SITELIB library contains system options and global information for your site. This library must exist before you call the %CPSTART macro. If it is a new empty library, then the SAS IT Resource Management application will initialize it the first time that it is specified on %CPSTART. *The default value for the SITELIB= parameter is set when SAS IT Resource Management is installed.*

If you specify the SITELIB= parameter but do not specify the SITEACC= parameter, then you will have WRITE (UNIX or Windows) or OLD (z/OS) access to the specified SITELIB library. If you do not specify the SITELIB= or the SITEACC= parameters, then the system provides READONLY (UNIX or Windows) or SHR (z/OS) access to the default SITELIB library.

SITESHR=*sitelib-server*
names the SAS/SHARE server that is used with SAS IT Resource Management to control access to SITELIB. This parameter is optional. Use this parameter only if you are using SAS/SHARE software with the SAS IT Resource Management solution at your site. A SAS/SHARE server enables multiple users to have update access to SAS data libraries. Caution should be exercised when you use SAS/SHARE in the SAS IT Resource Management environment, especially during periods when SAS data is being updated by batch processes. SAS/SHARE should be used only if multiple users need concurrent update access to SITELIB. To determine the value to use at your site, check with your site administrator or your SAS Installation Representative.

This parameter is not used for UNIX or Windows environments.

For more information about SAS/SHARE software, refer to the documentation for your current release of SAS.

%CPSTART Notes

Additional operating environment statements are required to wrap around %CPSTART and other macros in order to form a complete batch job stream. For more information, see “Duplicate-Data Checking” on page 671. Also, refer to the SAS Companion or other SAS documentation for your operating environment.

Calls to %CPSTART can run in true batch mode or can be submitted through the SAS Program Editor window. In either case, %CPSTART must be called within a SAS session. No other SAS IT Resource Management macro can be called and no SAS IT Resource Management macro variable can be used until you run the %CPSTART macro.

When SAS IT Resource Management is running, do not submit the %CPSTART macro through the SAS Program Editor window or you may get unpredictable results.

MXG

If, in this SAS IT Resource Management session, you intend to process data by using MXG-based staging code or to report on that data by using MXG formats, you must specify the MXGSRC= and/or MXGLIB= parameters. To check if your data uses MXG:

1. In the description of the %CPPROCES or %CMPROCES macro, see the description of the COLLECTR= parameter.
2. In the COLLECTR= description, see the Collector-Toolname Parameter Values table.
3. In the table, see the TOOLNM= column. If TOOLNM=MXG, processing uses MXG-based staging code.

   Also, if COLLECTR=GENERIC, you may decide to use MXG-based staging code if your data is collected by MXG.

If your data uses MXG, MXG must be accessible to the host on which this session runs.

Note: In a SAS IT Resource Management release, the tables and variables that are defined for MXG data correspond to a particular release of MXG. You can use the SAS IT Resource Management release with that MXG release. You can also use the SAS IT Resource Management release with any earlier or later MXG release that has the fields that you want to use. If you want to use fields in an MXG release that are not yet in a SAS IT Resource Management release, you can wait for the next SAS IT Resource Management release or you can create tables to correspond to any new MXG data sets that you want to use and create variables in existing tables to correspond to any new MXG data in existing MXG data sets.

For more information about tables, see the chapter ”Administration: Working with Tables” in the *SAS IT Resource Management User’s Guide*. For more information about variables, see the chapter ”Administration: Working with Variables” in the *SAS IT Resource Management User’s Guide*. △

Librefs

Library references are assigned as follows:

- *PGMLIB and SITELIB librefs*: these are always assigned.
- *CPSYSLIB libref*: this is only assigned only for the UNIX and Windows operating environments (that is, it is not assigned for z/OS).
- *PDB librefs (DETAIL, DAY, WEEK, MONTH, YEAR, ADMIN, DICTLIB, PDBWORK, and COLLECT)*:
  - In a batch session (specify MODE=BATCH on the %CPSTART macro), SAS IT Resource Management assigns one set of PDB librefs (that is, DETAIL, DAY, WEEK, MONTH, YEAR, ADMIN, DICTLIB, PDBWORK, and COLLECT) for one PDB when the %CPSTART macro executes.

    Note: In a batch session, if you want to switch PDBs, you have to issue a new %CPSTART. △

- In a GUI session (specify MODE=MENU on the %CPSTART macro or start the SAS IT Resource Management GUI using the SAS IT Resource Management GUI), SAS IT Resource Management assigns the PDB librefs (that is, DETAIL, DAY, WEEK, MONTH, YEAR, ADMIN, DICTLIB, PDBWORK, and COLLECT) for the PDB when you successfully activate the PDB.

  Note: In a GUI session, you can switch to another PDB and SAS IT Resource Management automatically assigns the new DETAIL, DAY, WEEK, MONTH, YEAR, ADMIN, DICTLIB, PDBWORK, and COLLECT librefs for the new PDB when you successfully activate the PDB.

  In a GUI session, multiple PDB librefs are associated with the activation of a specific PDB; they are not associated with a single %CPSTART. △

- *PDB libref*: this is assigned only when the PDB has MXG views specified.
- *RPGMLIB libref*: this is assigned if (and only if) connecting to a remote server. This applies to both MODE=BATCH and MODE=MENU for Windows and UNIX.
- *LIBRARY libref*: this is assigned if the MXGLIB macro parameter is specified when %CPSTART is invoked.

Note: %CPSTART assigns other librefs temporarily, as needed. △

%CPSTART Examples

Example 1 (UNIX)

To start the SAS IT Resource Management application in a batch job on UNIX and to specify the active PDB pathname, use the following statements. These examples use **pathname/pdbxxx** to indicate the name of a PDB.

```
%cpstart(mode=batch,
         pdb=pathname/pdbxxx,
         access=write,
         _rc=retcode);
%put cpstart return code= &retcode;
```

Note: Use the ACCESS=WRITE parameter only when you need to create or update the PDB. △

Example 2 (UNIX or Windows)

To start the SAS IT Resource Management application by using the SAS window interface with READ access to the same PDB that you used the last time that you used SAS IT Resource Management, type **itrm** on the SAS command line or submit the following statement from the SAS Program Editor window:

```
%cpstart();
```

Example 3 (z/OS)

To start the SAS IT Resource Management application in a batch job and to specify exclusive access to the active PDB, use the following statements:

```
%cpstart(mode=batch,
           root= root-location,
           pdb= pdb-name,
           disp=old,
           sitelib= site-library,
           siteacc=old,
           share= pdb-server,
           mxglib= mxg-mxg-formats,
           mxgsrc=('mxg-userid-source' 'mxg-mxg-source'),
           _rc=retcode
           );
%put CPSTART return code is &retcode;
```

By specifying SITEACC=OLD, you have the option of using the %CPBATCH macro, %CPHDAY macro, and %CPPDBOPT macro (if TYPE=SITELIB) later in this session. You must have write access to SITELIB in order to perform those tasks.

Example 4 (Windows)

This example starts a SAS IT Resource Management session on Windows and enables the data to be processed with MXG-based staging code later in the session:

```
%CPSTART(MXGLIB=c:\mxglib,
         MXGSRC=('c:\mxg\userid\sourclib' 'c:\mxg\mxg\sourclib'),
         PDB=my-pdb) ;
```

Example 5 (Parameters by Platform)

The following code calls %CPSTART:
For UNIX

```
%cpstart ( pdb=c:/temp/newpdb2
          ,access=write
          ,siteacc=write
          ,mode=batch
          ,_rc=cpstrc
         );
```

For Windows

```
%cpstart ( pdb=c:\temp\newpbd2
          ,access=write
          ,sitelib=c:\temp\sitelib
          ,siteacc=write
          ,root=c:\winsas\sas\addon\cpe
          ,mode=batch
          ,_rc=cpstrc
         );
```

For z/OS

```
%cpstart ( pdb=location.of.newpdb2
          ,disp=old
          ,siteacc=old
          ,mode=batch
          ,root=location.of.it.resource.management
          ,mxglib=location.of.mxg.mxg.library
          'mxgsrc=('location.of.mxg.userid.sourclib'
```

```
                'location.of.mxg.mxg.sourclib')
          ,_rc=cpstrc
    );
```

%CPTFORM

Builds a view that transforms data

%CPTFORM Overview

This macro builds a view (and optionally builds a data set) in which data is transformed in one or more of the following ways:

- by joining (or concatenating or interleaving) the data from two or more SAS views or SAS data sets
- by adding variables to the input data
- by adding variables to the output data.

The transformed data is especially useful for reports.

%CPTFORM Syntax

%CPTFORM(
IN1=*libref.name*
,**OUTVIEW=***libref.name*
<,**FORMULA=***(SAS-statement-list)*>
<,**IN2=***libref.name*>
.
.
.
<,**IN9=***libref.name*>
<,**JOINBY=***var-list*>
<,**JOINTYPE=***option*>
<,**OUTDATA=***libref.name*>
<,**PREFORM1=***(SAS-statement-list)*>
.
.
.
<,**PREFORM9=***(SAS-statement-list)*>
<,**_RC=***macro-var-name*>) ;

Details

IN1=,...,IN9=*libref.name*
specifies the input data sets or views to be transformed. The libref to this data set or view must already be defined by the time that this macro is called. *The IN1= parameter is required.* The IN2= through IN9= parameters are optional and, if specified, can be specified in any order. However, you should use the parameters sequentially. For example, if you are using only two input data sets or views, then use IN1= and IN2= (not IN1= and IN3=).

If more than one input parameter is specified, then the data is joined, concatenated, or interleaved. For more about joining, concatenating, and interleaving, see the JOINBY= and JOINTYPE= parameters.

OUTVIEW=*libref-name*
specifies the libref.*data-view-name* of the data view that is to be created. The view makes the transformed data available for use by other macros, such as the reporting macros. *This parameter is required.*

The libref for this location must already be defined by the time that this macro is called. The *data-view-name* must start with a letter and can contain letters and numbers. The name can be six characters long. If the data view does not exist, then it is created and used. If the data view already exists, then it is used and its contents are replaced by the new contents.

Other views, with names that are prefixed by OUTVIEW, can also be created by the macro in the same output library. Do not refer to these view names. They are referenced internally by other views.

Note: The value of this parameter is a complete SAS output data set specification and thus can contain SAS output data set options (such as WHERE=, INDEX=, and so on). To find more information about SAS output data set options, see the Help for your current release of SAS. △

FORMULA=(*SAS-statement-list*)
specifies a list of SAS statements to be included in the data view that is specified by the OUTVIEW= parameter (and implemented when the data set is generated, if the OUTDATA= parameter is specified).

The list of statements must be enclosed in parentheses, and each statement must end with a semicolon (;). This parameter is optional.

The FORMULA= variable is typically used to create formula variables in the output data view (and derived variables in the output data set, if OUTDATA= is specified).

CAUTION:
The syntax of these statements is not checked by %CPTFORM. Any syntax errors in these statements may cause the creation of the resulting view (and data set, if specified) to fail. △

JOINBY=*var-list*
specifies the list of one or more variables on which the join (or interleave) is based. If the list contains more than one variable, then separate the variables with one or more blanks. There are several cases:

- Only one input (IN1=) parameter is specified. In this case, the IN1= data is sorted by the *var-list*.
- More than one input parameter is specified and the JOINTYPE= parameter is set to CONCATENATE. In this case, this parameter is ignored.
- More than one input parameter is specified, the JOINTYPE= parameter is not set to CONCATENATE, and the JOINBY= parameter is specified. In this case, the variables in the *var-list* are used to join (or interleave) the observations.
- More than one input parameter is specified, the JOINTYPE= parameter is not set to CONCATENATE, and the JOINBY= parameter is not specified:
  - For any of the input data sets or views that are not in the active PDB, each one's BY variables list is obtained from the data set's or data view's SORTEDBY attribute, which is in the header in the SAS data set or view.
  - If the information is not available in the header, and the SAS data set or view is from the active PDB, then the BY variables list is obtained from

the corresponding BY variables list (if the input parameter's libref is DETAIL) or CLASS variables list (if the input parameter's libref is DAY, WEEK, MONTH, or YEAR) in the active PDB's data dictionary.

- If any list is missing or if the lists are not identical, then the macro terminates with a nonzero return code. Otherwise, the resulting list is used as if it had been specified by the JOINBY= parameter in the join (or interleave).

The macro uses SAS/SQL or a SAS DATA step to implement the join (or interleave). In SQL, these variables are used in the ORDER BY clause; in a SAS DATA step, these variables are used as the MERGE...BY list.

If any input data set (as defined by the input parameters) is not already sorted by these variables, then it will be sorted prior to joining. If any input data set or data view does not have all the variables that are specified in the *var-list*, then you can add the "missing" variables by using the corresponding PREFORM parameter.

JOINTYPE=*option*
specifies the type of join to perform. Options are INNER, LEFT, RIGHT, FULL, CONCATENATE, and INTERLEAVE, and *the default value is FULL*. If only one input parameter is specified, then the JOINTYPE= parameter is ignored. The valid values are as shown in this list, accompanied by a description of the simple case for each of the values.

INNER
Returns one combined observation for each observation in the input data set or view that has a matching observation in one or more of the other input data sets or views. If the first input observation has at least one match, then the resulting observation has values from variables on the original observation and from variables on the matching observation(s). If the first input observation does not have a match, then it is dropped. This is sometimes called an "inner join."

LEFT
Same as INNER, except that it returns one observation (combined or not) for each observation in the first input data set or view. If the first input observation has at least one match, then the resulting observation has values from variables on the original observation and from variables on the matching observation(s). If the observation does not have a match, then the resulting observation has values from the variables on the original observation and missing values for the variables that were not available from the other data sets or views. This is sometimes called an "outer join."

RIGHT
Same as LEFT, except that it uses the input data set or view that is specified by the parameter with the highest value of *n*, instead of the parameter with the lowest value of *n*. For example, if you specified IN1= through IN5= (but not IN6= through IN9=), then this join would use the input file that is specified by the IN5= parameter.

FULL
Same as LEFT, except that it also returns one observation for each observation in the IN2–9= data sets or views that did not have a match in the IN1= data set or view. As with LEFT, any variables whose values were not available from matched observations have missing values. This is sometimes called a "full outer join."

CONCATENATE

Input data sets or views are concatenated (as with a SAS SET statement) in the order that is listed in the IN1–9= parameters. Neither the input data sets or views nor the output data set and view are sorted. Observations are not combined.

INTERLEAVE
Input data sets or views are interleaved. For the order of interleaving, see the JOINBY= parameter. Observations are not combined.

For more information about these types, including information about data sets or views that have multiple matching observations, see the SQL procedure (PROC SQL) in the SAS documentation.

OUTDATA=*libref.name*
specifies the libref.*data-set-name* of the data set to be created. The data set makes the transformed data available for use by other macros, such as the reporting macros.

This parameter is optional. If it is not specified, then no data set is created. The libref must already be defined by the time that this macro is called. The *data-view-name* must start with a letter and can contain letters and numbers. The name can be eight characters long. If the data set does not exist, then it is created and used. If the data set already exists, then it is used and its contents are replaced by the new contents.

This option is a complete SAS output data set specification and thus can contain SAS output data set options (for example, WHERE=, INDEX=, and so on). For more information about SAS output data set options, see SAS documentation.

PREFORM1=,...,PREFORM9=(*SAS-statement-list*)
specifies a list of SAS statements that are to be implemented when data is acquired from the data set or view that is specified by the corresponding IN*= parameter. The list of statements must be enclosed in parentheses. Each statement must end with a semicolon (;). This parameter is optional.

The PREFORM parameters are typically used to create variables in the input data views and data sets. The variables are created before the data is sorted and before the data is joined or interleaved (if the data is to be joined or interleaved and if a sort is required before the join or interleave).

CAUTION:
The syntax of these statements is not checked by %CPTFORM. Any syntax errors in these statements might cause the creation of the resulting view (and data set, if specified) to fail. △

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CPTFORM Example

Example 1

In this example, each observation in the data view DETAIL.HN2IPI, in the active PDB, has the following variables:

- BY list variables: MACHINE IFINDEX DATETIME HOUR SHIFT
- metrics: INPKTS INERRS.

Additionally, each observation in the data view DETAIL.HN2IPO, in the active PDB, has the following BY list variables:

- BY list variables: MACHINE IFINDEX DATETIME HOUR SHIFT
- metrics: OUTPKTS OUTERRS.

The following code joins the observations in DETAIL.HN2IPI and DETAIL.HN2IPO and makes the result available from view ADMIN.HN2IP in the active PDB:

```
%CPSTART (..., pdb=pdb-name, disp=old or access=write, ...) ;

%CPTFORM( in1=detail.hn2ipi,
          in2=detail.hn2ipo,
          outview=admin.hn2ip,
          jointype=full );
```

The %CPSTART macro starts SAS IT Resource Management. The PDB that is specified by the PDB= parameter is the active PDB. This macro also defines some librefs, including the DETAIL libref, which refers to the DETAIL library in the active PDB, and the ADMIN libref, which refers to the ADMIN library in the active PDB. Additionally, the %CPSTART macro requests update/write access to the libraries in the PDB, because you must have update/write access in order to write to the ADMIN library.

The %CPTFORM macro creates a view that is named ADMIN.HN2IP in the active PDB. The view makes available the full-joined observations from DETAIL.HN2IPI and DETAIL.HN2IPO. Because the JOINBY= parameter is not specified, the macro checks in the active PDB's data dictionary. Because both DETAIL.HN2IPI and DETAIL.HN2IPO have the same BY variables list, that list is used as the basis of the join. Each observation that ADMIN.HN2IP makes available has the following variables:

- BY list variables: MACHINE IFINDEX DATETIME HOUR SHIFT
- metrics: INPKTS INERRS OUTPKTS OUTERRS.

Example 2

In this example, the input data view DETAIL.PCSPRO, in the active PDB, contains information about processes on UNIX. Each observation in DETAIL.PCSPRO has the following variables:

- BY list: MACHINE PCSUSER DATETIME HOUR SHIFT
- metrics: PCSCPTO (number of CPU seconds that the process uses) and MACHINE (the host name of the system that the process ran on).

The input data view MYLIB.EMPINFO, not in the active PDB, contains information on employees, including what each employee is charged for CPU use. Additionally, each observation in MYLIB.EMPINFO has the following variables:

- BY list variables: LASTNAME FRSTNAME
- metrics: DEPT RATE (cost of one CPU second) MACHINE ("Ford" or "Chevy").

The following code creates a view that is based on the observations in DETAIL.PCSPRO, with additional information from the RATE variable that is in MYLIB.EMPINFO:

```
%CPSTART( ..., pdb=pdb-name, disp=old or access=write, ... ) ;
LIBNAME mylib 'my-library-location' ;

%CPTFORM( in1=detail.pcspro,
          in2=mylib.empinfo (rename=(machine=carbrand)),
          outview=admin.usrpro,
          joinby=pcsuser,
          jointype=left,
          preform2=( length pcsuser $80.;
                     pcsuser = trim(dept) || lastname; ),
          formula=( label cost= "CPU cost";
                    format cost=dollar4.2;
                    if machine =: "Y2KLAB"
                       then cost = 0;
                       else cost = pcscpto * rate; ),
        );
```

The %CPSTART macro starts SAS IT Resource Management. The PDB that is specified by the PDB= parameter is the active PDB. %CPSTART also defines some librefs, one of which is the DETAIL libref, which refers to the DETAIL library in the active PDB, and another of which is the ADMIN libref, which refers to the ADMIN library in the active PDB. Additionally, this macro requests update/write access to the libraries in the PDB, because you must have update/write access in order to write to the ADMIN library. The LIBNAME statement defines MYLIB as the libref for my-library.

In the call to %CPTFORM, you see the following:

- The settings JOINTYPE=LEFT, IN1=DETAIL.PCSPRO, and OUTVIEW=ADMIN.USRPRO specify that the observations that view ADMIN.USRPRO will make available are based primarily on the observations in DETAIL.PCSPRO.
- The view DETAIL.PCSPRO already has a variable that is named PCSUSER, the SAS statements in the PREFORM2= parameter provide the data from MYLIB.EMPINFO with a variable that is named PCSUSER, and the JOINBY=PCSUSER setting specifies that PCSUSER is to be used to identify the observations to join.
- The specification for the IN2= parameters arranges the data from MYLIB.EMPINFO to identify the car information not by a variable that is named MACHINE, but by a variable that is named CARBRAND.
- The specification for the FORMULA= parameter creates a new variable COST that is to be available in the data from view ADMIN.USRPRO. Its value is based on the value of MACHINE, which is now only in the data from DETAIL.PCSPRO, on the value of PCSCPTO, which is in the data from DETAIL.PCSPRO, and on RATE, which is in the data from MYLIB.EMPINFO.
- As a result, view ADMIN.USRPRO makes available observations that correspond to the observations in DETAIL.PCSPRO. The ADMIN.USRPRO observations are made available with an additional variable, COST. In the observations that had no

match in MYLIB.EMPINFO, the value of COST is missing. In the observations that did have a match in MYLIB.EMPINFO, the value of COST is based on the calculation described by the FORMULA= parameter. The variables that are on the observations available from view ADMIN.USRPRO are:

BY List: PCSUSER (from JOINBY=)

Metrics:

- DEPT (from MYLIB.EMPINFO)
- RATE (from MYLIB.EMPINFO)
- CARBRAND (from MYLIB.EMPINFO)
- PCSCPTO (from DETAIL.PCSPRO)
- MACHINE (from DETAIL.PCSPRO)
- DATETIME (from DETAIL.PCSPRO)
- HOUR (from DETAIL.PCSPRO)
- SHIFT (from DETAIL.PCSPRO)
- COST (calculated from ADMIN.USRPRO's PCSCPTO, RATE, and MACHINE).

Example 3

This example includes three PDBs: PDB-name, PDB-A, and PDB-B. In this example, all three PDBs have a table that is named PCSGLB, and the definition of the table is the same in each of the PDBs.

The following example code interleaves the observations in view DETAIL.PCSGLB from all three PDBs and makes the results available from view ADMIN.ALLGLB in the active PDB.

```
%CPTFORM( in1=detail.pcsglb,
          in2=detailA.pcsglb,
          in3=detailB.pcsglb,
          outview=admin.allglb,
          jointype=interleave ) ;
```

The %CPTFORM macro creates a view that is named ADMIN.ALLGLB in the active PDB. The view makes available the interleaved observations from DETAIL.PCSGLB, DETAILA.PCSGLB, and DETAILB.PCSGLB. Because the JOINBY= parameter is not specified, the macro gets DETAIL.PCSGLB's BY list from the active PDB's data dictionary; it gets DETAILA.PCSGLB's BY list from its SORTEDBY attribute; and it gets DETAILB.PCSGLB's BY list from its SORTEDBY attribute. Because all three table definitions are the same, the lists are identical and the list is used to interleave the observations.

%CPUNCVT

Converts the active PDB from SAS IT Resource Management 2 format to SAS IT Resource Management 1 format

%CPUNCVT Overview

The %CPUNCVT macro converts a PDB from a format that can be used with SAS IT Resource Management 2 to a format that can be used with SAS IT Resource

Management 1. When you start SAS IT Resource Management 2 and open an existing PDB (one that is created or used with SAS IT Resource Management 1), the existing PDB is converted to a format that can be used with SAS IT Resource Management 2. If you need to convert that PDB back to the SAS IT Resource Management 1 format, then you can do so using this macro.

When you submit this macro, it converts the active PDB. You must have write or update access to the PDB that you want to convert. You can set the active PDB when you start SAS IT Resource Management (using the %CPSTART macro) or start the SAS IT Resource Management GUI, set the active PDB, and then submit this macro from the SAS Program Editor window.

%CPUNCVT Syntax

%CPUNCVT(
<**_RC=***macro-var-name*>);

Details

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CSATR2DD

Creates data dictionary control statements based on Aprisma SPECTRUM data

%CSATR2DD Overview

The %CSATR2DD macro creates dictionary control statements that can be used to create new table definitions for SPECTRUM model types that have not yet been defined to SAS IT Resource Management.

SAS IT Resource Management supplies table definitions for many common SPECTRUM model types. However, if a supplied table definition is not available for a model type whose data you want to use, then you can run this macro to define the

model type and create the new table definition. When you run this macro, %CSATR2DD creates %CPDDUTL CREATE TABLE control statements, and it also creates a CREATE VARIABLE control statement for each variable in the statistics data set. You can then run %CPDDUTL with those control statements to create a new table definition for your Aprisma SPECTRUM data.

To determine which data should be contained in the tables, %CSATR2DD reads the statistical data or the attribute data sets that are created by SPECTRUM Data Export utility.

%CSATR2DD Syntax

%CSATR2DD(
ATTR=*libref.name*
,**MODTYP=***libref.name*
,**EXTATTR=***libref.name* | **STAT=***libref.name*
<,**MAXNAMES=***number*>
<,**MODEL=***libref.name*>
<,**OUTFILE=***filename*>
<,**_RC=***macro-var-name*>
<,**WHICHAID=***attr-id-list*>
<,**WHICHMTH=***model-type-list*>
<,**WHICHTAB=**NEW | ALL>
<,**WHICHVAR=**ALL | NEW>);

Details

ATTR=*libref.name*
identifies the libref and name of the attribute data set that you exported from SPECTRUM. The %CSATR2DD macro uses this data set to associate the SPECTRUM attribute ID (in hex) from the STAT data set with the corresponding attribute name in this data set. The default data set name that is assigned by SPECTRUM Gateway is ATTRDESC; for Data Export it is ATTR. If you use one of these default names when you export your data, then use that same name as the value for the ATTR= parameter. If you do not use the default SPECTRUM name, then the value of the ATTR= parameter should be the name that you specified.

Specify the data set name by using the two-level format *libref.name*, where *libref* must be defined before calling this macro and *name* is the actual name of your attribute data set. The libref should point to the directory that contains the attribute data set that you exported from SPECTRUM.

For example, you might specify the following:

```
LIBNAME specgway
'/the/directory';
```

MODTYP=*libref.name*
identifies the libref and name of the model type data set that you exported from SPECTRUM. This data set associates SPECTRUM model type handles (in hex) from the STAT data set with a model type name in the data set that is specified by the MODTYP= parameter. The default name that is assigned by SPECTRUM Gateway and Data Export model type data set is modtyp. If you use this default name when you export your data, then use that same name as the value for the MODTYP= parameter. If you do not use the default SPECTRUM name, then the value of the MODTYP= parameter should be the name that you specified.

Specify the name by using the two-level format *libref.name*, where *libref* must be defined before you call this macro and *name* is the actual name of your model type data set. The libref should point to the directory that contains the model type data set that you exported from SPECTRUM.

For example, you might specify the following:

```
LIBNAME libref
'/the/directory';
```

EXTATTR=*libref.name*
identifies the libref and name of the extended attributes data set that you exported from SPECTRUM. The extended attributes data set enables %CSATR2DD to map each model type from the model type data set to its attribute IDs in the data set that is specified by the EXTATTR= parameter. The default data set name that is assigned by Gateway data is ATTROID, and for Data Export data it is EXTATTR. If you use one of these default names when you export your data, then use that same name as the value for the EXTATTR= parameter. If you do not use the default SPECTRUM name, then the value of the EXTATTR= parameter should be the name that you specified.

Specify the data set name by using the two-level format *libref.name*, where *libref* must be defined before you call this macro and *name* is the actual name of your extended attribute data set. The libref should point to the directory that contains the extended attribute data set. For example, you might specify the following:

```
LIBNAME libref '/the/directory';
```

Note: Specify either this parameter or the STAT= and MODEL= parameters. △

If you specify the EXTATTR= parameter, then the processing time might be lengthy, because specifying this parameter creates table definitions with all possible attributes for all model types that are defined for the SPECTRUM data base. By specifying the STAT= and MODEL= parameters instead, %CSATR2DD generates smaller tables that better correspond to the data that is being collected by SPECTRUM.

STAT=*libref.name*
identifies the libref and name of the data set that contains your logged data, which was exported from SPECTRUM. Each observation in this data set contains the values of the attributes that are logged at a given time for a given model. The default data set name that is assigned by SPECTRUM Gateway and Data Export is STAT. If you use this default name when you export your data, then use that same name as the value for the STAT= parameter. If you do not use the default SPECTRUM name, then the value of the STAT= parameter should be the name that you specified.

Specify the data set name by using the two-level format *libref.name*, where *libref* must be defined before you call this macro and *name* is the actual name of your statistics data set. The libref should point to the directory that contains the statistical data set that you exported from SPECTRUM. For example, you might specify the following:

```
LIBNAME libref '/the/directory';
```

Note: Specify either this parameter or the EXTATTR= parameter, but not both. If you specify the STAT= parameter, then you must also specify the MODEL= parameter. △

MAXNAMES=*number*
indicates the maximum number of attributes that are permitted per model type. *The default value is 512*. If you have more than 512 attributes for each model type, then change this value to the appropriate number.

MODEL=*libref.name*
identifies the libref and name of the model data set that you exported from SPECTRUM. The %CSATR2DD macro uses this data set to associate the model handles (in hex) from the data set that is specified in the STAT= parameter with the model or host names in the data set that is specified in the MODEL= parameter. The model name is used as the value for the MACHINE variable in SAS IT Resource Management.

The default data set name that is assigned by both Gateway and Data Export is MODEL. If you use that default name when you export your data from SPECTRUM, then specify that same default name as the value for the MODEL= parameter. If you specify a name other than that default name, then use that name as the value of the MODEL= parameter.

Specify the data set name by using the two-level format *libref.name*, where *libref* must be defined before you call this macro and *name* is the actual name of your model handle data set. The libref should point to the directory that contains the model handle data set that you exported from SPECTRUM. For example, you might specify the following:

```
LIBNAME libref '/the/directory';
```

Note: You must specify this parameter if you specify the STAT= parameter. This parameter CANNOT be specified if the EXTATTR= parameter is specified. △

OUTFILE=*filename*
identifies the path and name of the output file that is created by calling this macro. The file contains the %CPDDUTL control statements that define the PDB tables and variables, which correspond to the STAT data set or SPECTRUM data sets for %CSATR2DD and correspond to a SunNet Manager schema file for %CSSNMSCH.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

WHICHAID=*attr-id-list*
identifies the attribute IDs that should be included in the table definition. This parameter enables you to subset your data so that the resulting table definition reflects only meaningful attributes. The attributes that are listed here are included in the table definitions that are created on the basis of the model types that are specified in the WHICHMTH= parameter. You can identify the attributes that should be included in the table definition as shown in the following example:

```
WHICHAID=attrid in (65134 65980
64900)
```

In this example, only the variables with the specified attribute IDs are included in the table definition. *If you do not specify a value for this parameter, then all attribute IDs are included in the table definitions*.

WHICHMTH=*model-type-list*
identifies a list of model types for which table definitions should be created. A table definition is created for each model type listed here. Each table definition includes the attributes that are listed in the WHICHAID= parameter. This parameter enables you to subset your data and create table definitions for a selected list of model types, as shown in the following example.

```
WHICHMTH=mth in (65134 65132)
```

In this example, table definitions are created only for the selected model type handles (mth's): 65134 and 65132.

If you do not specify a value for this parameter, then table definitions are created for all model types.

WHICHTAB=NEW | ALL
identifies the tables for which table definitions should be created. This parameter creates tables that are based on the value of the WHICHMTH= parameter. To create %CPDDUTL CREATE TABLE statements for all model types that are encountered in the SPECTRUM data sets, set WHICHTAB=ALL. *To create %CPDDUTL CREATE TABLE statements for new tables only, leave this parameter set to the default value, WHICHTAB=NEW*. NEW indicates that new table definitions are created only for the model types that do not have corresponding table definitions within SAS IT Resource Management.

WHICHVAR=ALL | NEW
identifies the variable definitions that should be included in the table definition. You should create a variable for each attribute that you want to include in the table definition. This parameter enables you to subset the data so that only selected variables are included in the table definition.

To generate %CPDDUTL CREATE VARIABLE control statements for all attributes in the logged data, leave this parameter set to the default value, WHICHVAR=ALL. To generate %CPDDUTL CREATE VARIABLE control statements only for attributes that have not been defined in any table definition, set WHICHVAR=NEW.

%CSATR2DD Notes

The file that is specified by the OUTFILE= parameter contains the CREATE TABLE and CREATE VARIABLE control statements and can be used directly with the %CPDDUTL macro. *If you do not provide a value for this parameter, then the output is displayed in the SAS log*.

%CSATR2DD Example

This example generates a table definition that contains all the attributes that are exported into *specgway.stat* and lists the table definition in the SAS LOG window.

```
*Define the specgway libref;
libname specgway '/usr/spectrum/export.output' ;
```

```
*Generate table definition statements;
%csatr2dd(modtyp=specgway.modtyp,
          attr=specgway.attr,
          model=specgway.model,
          stat=specgway.stat,
          outfile=/tmp/table.ddutl,
          whichtab=ALL,
          whichvar=ALL ) ;

*Read table definition statements into %CPDDUTL;
%cpddutl (filename=/tmp/table.ddutl, list=y ) ;
```

%CSCSIFMT

Builds site-specific formats for Aprisma SPECTRUM data on UNIX

%CSCSIFMT Overview

The %CSCSIFMT macro creates SAS formats that associate the Aprisma SPECTRUM model handles (the host IDs used by the SAS IT Resource Management application) with the model names of the hosts. These formats are used to identify the data when you run %CSPROCES to process the data.

The host names that are used by SPECTRUM Data Gateway software are stored in the model data set. This data set, which SPECTRUM names model by default, is read by %CSCSIFMT using the MODELDS= parameter. The %CSCSIFMT macro then creates the formats based on the model data set and stores them in your SITELIB directory. You must have write access to your site's SITELIB directory in order to run this macro.

The first time that you process SPECTRUM data and any time that you change a host name in your network, you must follow these steps:

1. Rebuild the model data set by running SPECTRUM Data Gateway software.
2. Run %CSCSIFMT in order to create the necessary SAS formats in SITELIB.

Perform these steps before running %CSPROCES on the data set that includes the new model handles.

%CSCSIFMT Syntax

%CSCSIFMT(
,<**MODELDS=***libref.name*>
<**,MODELFIL=***physical-filename*>
<,**_RC=***macro-var-name*>);

Details

MODELDS=*libref.name*
specifies the libref and name of the SPECTRUM model data set that was created by the SPECTRUM Data Gateway software. The libref must be defined before this macro is called. The value for this parameter is usually *libref*.MODEL. You can use a SAS LIBNAME statement to associate the *libref* with the SPECTRUM Data Gateway SAS data library, as illustrated here:

```
libname specgway
'pathname/gateway';
```

MODELFIL=*physical-filename*
is the physical location and filename of a model handle file that is stored as an external file. This includes the complete directory path and the filename. Specify either MODELFIL= or MODELDS=, but not both. Either form of the model data set is acceptable.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

%CSCSIFMT Notes

Format definitions are stored in memory. When you create a format definition in batch with %CSCSIFMT, you can use the format immediately. However, when you use %CSCSIFMT to redefine a format definition while the SAS IT Resource Management GUI for this application is running, you must restart the SAS IT Resource Management GUI before using the updated format definition.

%CSCSIFMT Example

This example updates your SITELIB library with the formats for network host names and model types and then processes SPECTRUM data.

```
%cpstart(mode=batch,
         pdb= pathname/pdbxxx,
         access=write,
         sitelib= sitelib-to-update ,
         _rc=retcode);
%put cpstart return code= &retcode;

   /* assign libref for SPECTRUM data */
libname specgway 'pathname/gateway';

   /* create SAS formats in SITELIB library */
%cscsifmt(modelds=specgway.model);
```

```
%put cpstart return code= &retcode;


   /* process SPECTRUM data  */
%csproces(specgway.stat,
          cciifp cxytse csibce csil23 csihi2 csipng cciigs,
          collectr=spectrum,
          _rc=retcode);
%put csproces return code is &retcode;
```

%CSPROCES

Processes data into the detail level of the active PDB on UNIX

%CSPROCES Overview

The %CSPROCES macro runs the process task, which includes reading and validating performance data and transforming the performance data from its original logged files into the detail level of the active PDB. In addition, %CSPROCES updates data dictionary entries that describe the amount of data in the PDB and the most recent date and time of processing data into the detail level of the tables.

%CSPROCES Syntax

%CSPROCES(
<*rawdata*>
,*tablist*
,**COLLECTR=***collector-name*
,**TOOLNM=***tool-name*
<,**AGELIMIT=***number*>
<,**COLDUMP=***command*>
<,**COLLVER=***version-number*>
<,**DELIM=***'delim_chars'*>
<,**EXITSRC=***exit-source*>
<,**GENLIB=***libref*>
<,**GROUP=***group-file*>
<,**LOGFMT=**NORMAL | MULTITAB | SINGLETAB>
<,**MAXDATE=***date*>
<,**MINDATE=***date*>
<,**OPTIMIZE=**TIME | DISK>
<,**PASSWD=***password-file*>
<,**_RC=***macro-var-name*>
<,**RSHHOST=***remote-command*>
<,**SUBSET=***expression*>);

Details

rawdata
specifies the name of the raw data file. The name includes the complete directory path and name of the raw data file to be processed. This positional parameter is

not required, but if it is specified, it must be first. If you omit this parameter, then you must use a comma as a placeholder. The name can be any of the following:

- the name of the raw file that contains data from your data collector (for example, HP Network Node Manager or IBM Tivoli NetView). The name of the file can be enclosed in quotation marks, but quotation marks are not required.
- a PIPE command that sends the raw file records to standard output
- a SAS data set that contains SPECTRUM data, as created by the Spectrum Gateway for SAS. This data set is typically the "stat" data set that is created by the gateway. Specify this name in the form

  ```
  libref.dataset
  ```

  If the libref is WORK, you can omit the *libref* and period.

Omit the *rawdata* parameter under the following circumstances:

- if you want to reprocess existing data in the detail level by using the AGELIMIT= or SUBSET= parameters. For more information, see the AGELIMIT= and SUBSET= parameters.
- if you are processing data by using COLLECTR=GENERIC. In this case, specify the GENLIB= parameter instead of *rawdata*.

When you process UNIX accounting data (collected by the *accton* command), you must edit the data file and add a line at the beginning of the file that contains the host name:

```
machine= hostname
```

where *hostname* is the name of the machine that created the data. Or you can use a *pipe* command to prefix the host name to the data file (see the examples for this macro). This information is used to set the MACHINE variable in the PDB data libraries.

tablist
lists one or more tables into which the logged data should be processed. *This positional parameter is required* and must be specified in the second position. If you specify more than one table, then use blanks to separate the tables in the list.

For HP OpenView Performance Agent input files, you can specify an asterisk (*) as the value of the tablist parameter if you want to process data from the raw data file into the appropriate tables.

COLLECTR=*collector-name*
specifies the name of the data collector or data source that provides the raw data. The value that you specify for the COLLECTR= parameter depends on which process macro you use. The following example uses %CSPROCES with the SUNETMGR collector:

```
%CSPROCES( ...collectr=sunetmgr,...);
```

Also, the TOOLNM= parameter may be required depending on which value you specify for the COLLECTR= parameter. The following example uses %CWPROCES with the WEBLOG collector:

```
%CWPROCES( ...collectr=weblog, toolnm=clf,...);
```

See the following COLLECTR - TOOLNM Parameter Values table for valid values of the COLLECTR= parameter and the TOOLNM= parameter. In the COLLECTR= Value and TOOLNM= Value columns of this table, the following process macro abbreviations apply:

%CM for %CMPROCES

%CP for %CPPROCES
%CS for %CSPROCES
%CW for %CWPROCES.

Table 2.11 COLLECTR - TOOLNM Parameter Values

COLLECTR= Value	Collector Description	TOOLNM= Value
GENERIC for %CM, %CP, %CS, and %CW	data collector or data source for which SAS IT Resource Management does not supply table definitions and the user has not installed table definitions	SASDS for %CM SASDS or CHARDELIM for %CP, %CS, and %CW
ACCUNX for %CP	UNIX accounting	SASDS
DCOLLECT for %CM	IBM DFSMS Data Collection Facility	MXG
EREP for %CM	IBM SYS1.LOGREC Error Recording	MXG
ETEWATCH for %CP	End-To-End Watch (transaction response measurement) by Candle	ETE12 (ETEWatch 1.2 from Candle) ETE13 (ETEWatch 1.3 from Candle) EBA (EBA logs from Candle)
HP-MWA for %CS and %CW	HP MeasureWare	MWA-BE or MWA-LE (See Note 6 following the table)
HP-OV for %CS and %CW	IBM Tivoli NetView or HP Network Node Manager	not applicable
HP-OV-C for %CS	IBM Tivoli NetView or HP Network Node Manager + Coerce	not applicable
HP-OVPA for %CS and %CW	HP OpenView Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
HP-PCS for %CS and %CW	HP Performance Collection Software	MWA-BE or MWA-LE (See Note 6 following the table)

COLLECTR= Value	Collector Description	TOOLNM= Value
HP-PCS for %CP	HP OpenView Reporter	SASACCESS
HP-VPPA for %CS and %CW	HP VantagePoint Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
IMF for %CM	Boole & Babbage IMF from BMC	MXG
LSPW for %CS and %CW	Landmark Performance Works	CHARDELIM
NETCONV for %CP	Cisco IOS NetFlow	NETFLOW
NTSMF for %CM and %CP	Windows with Demand Technology's NTSMF product	MXG for %CM SASDS for %CP
PATROL for %CP	BMC Patrol	SASDS
PROBEX for %CS	Probe/X by Landmark in SunNetMgr format	not applicable
ROLMPBX for %CP	Rolm PBX	SASDS
SAPR3 for %CP	SAP R/3	SASADAPT (SAS IT Management Adapter for SAP) STATBIN (Statistics File) STATRFC (Using SAP R/3 API) (See Note 7 following the table)
SAR for %CP	UNIX sar command (system activity reporting)	SASDS
SITESCOP for %CP	SiteScope by Mercury Interactive	SASDS

COLLECTR= Value	Collector Description	TOOLNM= Value
SMF for %CM and %CP	IBM z/OS System Management Facilities	MXG
SPECTRUM for %CS	SPECTRUM by Aprisma	not applicable
SUNETMGR for %CS	SunNet Manager and Enterprise Manager	not applicable
TMON2CIC for %CM	Landmark "The Monitor for CICS/ESA 2"	MXG
TMONCIC8 for %CM	Landmark "The Monitor for CICS" (older)	MXG
TMONCICS for %CM	Landmark "The Monitor for CICS" V 1.3 and later	MXG
TMONDB2 for %CM	Landmark "The Monitor for DB2"	MXG
TMS for %CM	CA-TMS (Release 5 or later)	MXG
TPF for %CM	IBM Transaction Processing Facility	MXG
TRAKKER for %CS	Trakker by Concord Communications	not applicable
VISULIZR for %CP	Visualizer by BMC Software	SASACCESS
VMMON for %CM	VM Monitor	MXG

COLLECTR= Value	Collector Description	TOOLNM= Value
WEBLOG for %CP	Web server log	CLF (Common Log Format) ELF (Extended Log Format) IISORIG (Microsoft IIS original Weblog format) WEBETE (ETEWatch Web browser logs from Candle) WEBEBA (EBA logs from Candle)

1 *For COLLECTR=GENERIC:* For more information about using the Generic Collector Facility, see the chapter "Setup Case 3" in the *SAS IT Resource Management User's Guide* and the chapter "Setup Case 4" in the *SAS IT Resource Management User's Guide*, and the chapter "Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

   For more information about installing user-written or consultant-written table definitions, see "%CPPKGCOL" on page 144, "%CPRPTPKG" on page 177, and "%CPINSPKG" on page 124.

2 *For %CMPROCES:* The COLLECTR= parameter is required. The TOOLNM= parameter is also required.

3 *For %CPPROCES:* The COLLECTR= parameter is optional. If the parameter is not specified, then the default is COLLECTR=GENERIC. The TOOLNM= parameter is also optional. If the parameter is not specified, then the default is TOOLNM=SASDS.

4 *For %CSPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

5 *For %CWPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

6 *For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA:* The TOOLNM= parameter is used only when you use a PIPE command for the raw data.

   For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA, the TOOLNM= value is used to identify the endian type of the data. For example, data from Windows platforms is little-endian and data from most UNIX platforms is big-endian.

   Specify TOOLNM=MWA-BE if the incoming data is big-endian (BE). Specify TOOLNM=MWA-LE if the incoming data is little-endian (LE).

   If you use a PIPE command for the raw data and you do not specify the TOOLNM= parameter, SAS IT Resource Management attempts to determine the endian type of the incoming data. If the attempt is successful, the processing step continues. If the attempt is not successful, the processing step terminates with a message.

   Note: HP OpenView Performance Agent (HP-OVPA) is the latest name for these products: HP-PCS, HP-MWA, and HP-VPPA. △

7 *SAP:*

   □ *For SAP Release 4.6D and earlier, use TOOLNM=STATBIN or TOOLNM=STATRFC.*

      You must set one macro variable before you run %CPPROCES. Additionally, there are four other optional variables that you might need to set, depending on the data that you are processing.

The required macro variable is

- SAPVER - the SAP R/3 version.

The optional macro variables are

- SAPSYSNR - the SAP R/3 system number
- SAPSYSNM - the SAP R/3 system name
- SAPHOST - the SAP R/3 application server name
- SAPPLAT - the platform on which the SAP STAT file was generated.

You need to specify the SAPPLAT macro variable only if you are processing SAP STAT data that was generated in a Windows operating environment. The default value for the SAPPLAT macro variable enables you to read data that is generated in both UNIX and z/OS operating environments. However, if you read data that was generated in a Windows operating environment, then set SAPPLAT to a value of WIN. (For more information, see the examples for the %CPPROCES macro.)

You must specify the SAPSYSNM, SAPHOST, and SAPSYSNR macro variables only if you use the SAP R/3 collector and only if you process a single data file.

To populate these macro variables automatically, see the instructions in the RAWDATA= parameter for the %CPPROCES macro.

- *For SAP releases later than 4.6D, use TOOLNM=SASADAPT.*

  You can also use TOOLNM=SASADAPT with some earlier releases of SAP.

8 *If you are working with a Candle collector:*

- Use COLLECTR=ETEWATCH, TOOLNM=ETE12 or ETE13 or EBA when application and transaction analysis is desired.
- Use COLLECTR=WEBLOG, TOOLNM=WEBETE or WEBEBA when processing WebBrowser behavior module data or EBA data for which a hierarchical page analysis (for example, jobs —> Cary —> R&D) is desired.

9 *For COLLECTR=SMF on UNIX or Windows:* See "Example 8: Processing SMF data on UNIX or Windows" in the %CPPROCES Examples.

10 Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

TOOLNM=*tool-name*
typically specifies the name of the software that will be used to transform the raw data and to stage the data into SAS data sets or data set views. The value that is specified for this parameter is based on the value that you specify for the COLLECTR= parameter.

For more information about values of the TOOLNM= parameter, see the COLLECTR= parameter. The COLLECTR= parameter contains a table of COLLECTR= and TOOLNM= combinations and also contains collector-specific and macro-specific notes.

Note: Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*. △

AGELIMIT=*number*
is the number of days' worth of data to keep in the detail level of every table in the PDB. *The default is the value that is defined in the data dictionary for each table's detail level*. Use this parameter to override, on a specific run, the age limit that is specified in the data dictionary for a table in the PDB. When you specify this parameter, it does not reset the dictionary value.

If you do not specify input by using the *rawdata* parameter or the GENLIB= parameter, then this macro reprocesses any existing data in the detail level (and possibly ages out old data), based on the values of the AGELIMIT= and SUBSET= parameters.

Note: If AGELIMIT=0 at the detail level in the data dictionary, then your data is stored in the detail level until the next reduce or process task runs. Therefore, if the reduce task runs immediately after the process task, then the data is processed, reduced into the summary levels, and then removed from the detail level. However, if the process task runs again before the data has been reduced, then the existing data in the detail level is deleted before it is reduced to the summary levels. Using AGELIMIT=0 is useful if you are collecting and processing large amounts of data on a daily basis, but to avoid losing data you should always reduce the data immediately after it is processed. △

Note: For information about setting age limits for your data, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*. △

COLDUMP=*command*
specifies the name of the snmpColDump command to be used to dump HP-OV data at your site. *The default value for UNIX is /opt/OV/bin/snmpColDump and, by default, this command is installed in the directory where HP-OV is installed. The default for Windows environments is snmpColDump.*

For Windows environments, the value for this parameter is simply the snmpColDump command. You do not have to specify the full path for the location of the command. However, your path must contain the location of the snmpColDump utility.

COLLVER=*version-number*
specifies the version number of the SunNet Manager collector from which you are processing data. This parameter is used to process data from older versions of the collector. Specify this parameter only if the data was collected using SunNet Manager Version 1, and in this case use the value COLLVER=1. Otherwise, do not specify this parameter.

DELIM=*'delim_chars'*
specifies the character(s) that is used as a value separator in the character-delimited data. The delimiter character or list of delimiter characters must be enclosed in matched single or double quotation marks. Do not use a blank space to separate the items in the list. Specify a blank space in the list only if a blank space is one of the delimiters in your file.

If you do not specify a delimiter and you are processing character-delimited data, then the default value, a blank, is used.

Note: This parameter takes effect if TOOLNM=CHARDELIM or if COLLECTR=NTSMF.

EXITSRC=*exit-source*

specifies the location where exit routines (source programs) for SAS IT Resource Management are stored. The value of this parameter can be the name of a SAS catalog that is specified in the form of *libref.catname*, or it can be a specified location in your operating environment. For UNIX or Windows, this is the complete pathname of a directory. For z/OS, this is the fully qualified name of a partitioned data set.

If you specify a SAS catalog name, then use a SAS LIBNAME statement in order to associate the *libref* with the SAS library that contains the catalog.

If you want to process your data without using exit routines, then do not specify this parameter.

For the %CSPROCES and %CWPROCES macros, observations with duplicate values for the BY list variables are removed by default. If you want to prevent the removal of duplicate observations, use the PROC180 exit point that is described in "Exit Points for %CSPROCES and %CWPROCES" in the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

For more information on exit processing, see the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

Note: For the %CSPROCES and %CWPROCES macros, if you use process exits and you process HP OpenView Performance Agent data, then you must specify LOGFMT=SINGLETAB in order for your data to be processed. For more information, see the LOGFMT= parameter. △

GENLIB=*libref*
specifies the *libref* for the SAS library that contains the staged data that is to be processed into the PDB. The staged data has been created by user-provided software and is to be processed into a PDB by using COLLECTR=GENERIC. The data may be a SAS data set, a SAS DATA step view, a SAS SQL view, or a SAS/ACCESS view. For each table, the name of the data set, DATA step view, SQL view, or SAS/ACCESS view that has staged data for the table is specified by the table's external name parameter.

Use this parameter to specify the input data set or view for the SAS IT Resource Management process macro that you use to process your data, but only if you specify COLLECTR=GENERIC and TOOLNM=SASDS. If you specify this parameter for the %CPPROCES macro, then do not specify the RAWDATA= parameter. If you specify the GENLIB= parameter for the %CMPROCES macro or the %CSPROCES macro or the %CWPROCES macro, then do not specify the (positional) *rawdata* parameter.

GROUP=*group-file*
specifies the /etc/group file that can be used to translate *accton* process data group IDs from numbers to text strings. You must specify the group file from the host that collected the accounting data. *The default value is /etc/group*.

This parameter is used only with COLLECTR=ACCTON.

LOGFMT=NORMAL | MULTITAB | SINGLETAB
indicates whether special handling is required to process the input data log with %CSPROCES or %CWPROCES. *The default value is MULTITAB. NORMAL is an alias for MULTITAB.* This parameter is used only if COLLECTR=HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA.

If LOGFMT=MULTITAB or NORMAL and COLLECTR=HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA, then the SAS IT Resource Management process macro reads the input file that contains metrics from multiple tables or different metrics from the same table. The input file might contain data from multiple hosts that are concatenated together, but it must be a flat file (not data that is piped from the

HP OpenView Performance Agent extract command). The data that is used with LOGFMT=MULTITAB or NORMAL must be clean data; that is, the data must not contain error or status messages between the records that are sometimes produced by extract when piping its output to a file.

If messages indicate that you need to specify a value with LOGFMT=MULTITAB, then use the format **LOGFMT=MULTITAB:nnnnn**, where **nnnn** is the maximum number of item IDs that can be processed. Normally the default value (10,000) will suffice.

If LOGFMT=SINGLETAB and COLLECTR=HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA, then the SAS IT Resource Management process macro requires the input data file to contain metrics from only one table (for example, GLOBAL, DISK, PROCESS, and so on) and precisely the same metrics must be collected from each occurrence of the table in the input file (data from different tables or different metrics from the same table are skipped and messages about the skip are written to the SAS log). However, the input file can contain data from multiple hosts that are concatenated together and can be either a flat file or piped output from the HP OpenView Performance Agent extract command.

Note: HP-OVPA is the latest name for the collectors HP-PCS, HP-MWA, and HP-VPPA. △

You must specify LOGFMT=SINGLETAB if

- you are using PIPE to read raw data with %CSPROCES
- you are processing HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA data and you are using process exits (by using the EXITSRC= parameter).

MAXDATE=*date*

specifies the maximum acceptable date in incoming (new) data. The date value must use a valid SAS date-constant format. The format for valid SAS dates requires that the date be enclosed in quotation marks. For example,

```
'01-JAN-03'd
```

is an example of a valid SAS date for January 1, 2003, and

```
'01-JAN-2001'd
```

is an example of a valid SAS date for January 1, 2001. Do not use datetime constants such as

```
'01-JAN-02:23:30'dt
```

The following example shows how to use the MAXDATE= parameter to process data that has values for the DATETIME variable that are as far away as 10 days in the future. (Other parameters can be included as required.)

```
%CSPROCES(..., maxdate=TODAY()+10, ...) ;
```

The following example shows how to use the MAXDATE= parameter to process data up through the date of December 31, 2002. (Other parameters can be included as required.)

```
%CSPROCES(..., maxdate='31Dec2002'd, ...) ;
```

If the incoming data contains a DATETIME value that is greater than the value that you specify for this parameter, then processing stops before any new data is written to the detail level. *The default value is TODAY()+2*, which means that if any of the input data is dated more than 48 hours after the system date on the system where the data is being processed, then processing will stop.

MINDATE=*date*

specifies the minimum acceptable date in incoming (new) data. The date value must use a valid SAS date-constant format. The format for valid SAS dates requires that the date be enclosed in quotes. For example,

```
'01-JAN-02'd
```

is an example of a valid SAS date for January 1, 2002, and

```
'01-JAN-2001'd
```

is an example of a valid SAS date for January 1, 2001. Do not use datetime constants such as

```
'01-JAN-03:23:30'dt
```

If the incoming data contains a DATETIME value whose date is earlier than the value that you specify for this parameter, then processing stops. The default value is TODAY()-3650, which means that data up to 10 years old can be processed into the PDB.

OPTIMIZE=TIME | DISK
specifies whether time or disk space is more important to you when processing data. The default value is TIME. Do not specify this parameter if you specify COLLECTR=GENERIC.

When merging new data into the detail level of the PDB, SAS IT Resource Management creates a new data set, merges into it the incoming data and the data from the old data set, and then deletes the old data set. If the OPTIMIZE= parameter is set to (or defaults to) TIME, then the new data set is created on the same disk as the old data set. However, that disk may not have room to hold (temporarily) both the new and the old data sets. If that is the case, then specify OPTIMIZE=DISK, which causes the new data set to be created on the disk where the SAS WORK library resides and, at the end of the process step, to be copied to the disk where the old data set resided.

Note: When you specify OPTIMIZE=DISK, the copy of the table is stored in the library to which the libref WORK refers. If you do not want to use this library and disk, then use the **-work** SAS system option to change the WORK libref to refer to a different library. Be sure that the PDB and the WORK directory that you specify are on disks with sufficient space to store the largest detail table that you will process. △

If you are backloading data, then the space may need to be larger. For more details on the size of PDBs, refer to the "%CxPROCES Macros Recovery Procedures" in "Troubleshooting Batch Jobs That Fail" on page 655.

Note: %CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES. △

PASSWD=*password-file*
specifies the full path and name of the /etc/passwd file that can be used to translate *accton* user IDs from numbers to text strings. You must specify the password file from the host system that collected the accounting data. *The default value is /etc/passwd.*

This parameter is used only with COLLECTR=ACCTON.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

RSHHOST=*remote-command*
specifies the command and host to use in submitting other commands for remote execution, if you are using the HP-OV collector. For example, you might specify

```
rshhost=remsh netmon1
```

where **remsh** is the UNIX command for executing remote commands and **netmon1** is the remote host on which to execute the commands. The value of RSHHOST= is used as the prefix for the value of the COLDUMP= parameter.

This parameter is used only with COLLECTR=HP-OV or HP-OV-C.

SUBSET=*expression*
specifies a data-subsetting expression or filtering criteria for new data that is being processed into the detail level and for existing data that is already stored in the detail level. You cannot use this parameter if you specify COLLECTR=GENERIC.

If you specify the *rawdata* parameter and if the expression is true, then the observation is stored in the detail level. If the expression is false, then the observation is not stored in or is deleted from (for currently stored data) the detail level. If you do not specify the *rawdata* parameter, then SUBSET= affects only the data that is currently stored in the detail level of the PDB.

The format of the value *expression* follows the rules for IF expressions. Refer to *SAS Language Reference* documentation for your current version of SAS to find details on specifying expressions.

%CSPROCES Notes

If you have made dictionary updates that require table views to be rebuilt, such as by adding variables, derived variables, or formula variables, then the views will be rebuilt automatically when %CSPROCES runs. For more information on updating tables in batch, see %CPDDUTL.

If you are using the HP-OV collector and you plan to use the UNIX command **remsh** to execute commands on remote hosts, then you must have write access to the *rawdata* file that is specified on %CSPROCES.

The MINDATE= and MAXDATE= parameters on %CSPROCES enable you to control whether future data is processed into the PDB. By default, these parameters prevent future data from being inadvertently processed into a PDB and thus from subsequently aging older data out of the PDB. However, for COLLECTR=GENERIC on %CSPROCES, this capability is implemented by using the CPFUTURE global macro variable instead of by using the MINDATE= and MAXDATE= parameters.

If you specify COLLECTR=SPECTRUM, then you must run the %CSCSIFMT macro before you process your data for the first time. Otherwise, the %CSPROCES macro will fail.

The PROBENET, SUNETMGR, HP-OVPA, HP-VPPA, HP-MWA, and HP-PCS collectors present time information in Coordinated Universal Time (formerly known as Greenwich Mean Time). Therefore, if you specify one of these collectors, then you must specify the distance (in time zones) of your site from Greenwich, England, before you process your data. This can be done through the SAS IT Resource Management GUI for UNIX or Windows by selecting the **Manage PDBs** task on the Administration tab. Select Properties from the Manage PDBs window to set options for the active PDB. If you do not specify this value, then SAS IT Resource Management will use the value that is supplied in the header record of the raw data. If the value is not supplied in the header, then SAS IT Resource Management will use the value from the active PDB's properties.

Note: By default, the %CSPROCES macro deletes duplicate observations, that is, those observations that have identical values in all of the BY or CLASS variables. If you want to prevent the deletion of duplicate observations, use the EXITSRC= parameter and the PROC180 exit. For more information, see the EXITSRC= parameter. △

%CSPROCES Examples

Example 1: Using Multiple Time Zones

This example processes data from one file by using the default time zone for the PDB (the Eastern time zone of the United States) and then resets the time zone macro variable to process data from a different time zone. These values can be set by using the SAS IT Resource Management GUI or by using the **cpgmtdev** global macro variable, as demonstrated in this example.

```
/* this processes the local data */
%csproces('/snm/cityname/logfile',
          snmhpf,
          collectr=sunetmgr,
          _rc=retcode);
%put return code from Cityname csproces is &retcode;
  /* this sets up to process data from a different time zone*/
%let olddst=;  /* save previous Daylight Saving Time */
%let olddev=;  /* save previous time zone */
%let cpdstime=;   /* no Daylight Saving Time */
%let cpgmtdev=-1; /* specifies a city one hour east of Greenwich */
    /* process the new daylight savings time data */
%CSPROCES('/snm/cityname/logfile',
          snmhpf,
          collectr=sunetmgr,
          _rc=retcode);
%put Return code from other CSPROCES is &retcode;
%let cpdstime=;  /* restore previous Daylight Saving Time */
%let cpgmtdev=;  /* restore previous time zone */
```

For additional information, see
the section "Handling Time Stamps for Coordinated Universal Time" in the chapter "Administration: Working with Data" in the *SAS IT Resource Management User's Guide*.

Example 2: %CSPROCES for TRAKKER

This example processes TRAKKER data. The name of the raw data file in the %CSPROCES macro is enclosed in quotation marks, but the quotation marks are not required.

```
%csproces('/disk1/tkr/rawdata/concord.9302.rawdata',
          tkrpcl tkrsts tkrip_ tkrdll,
          collectr=trakker,
          _rc=retcode);
%put csproces return code= &retcode;
```

Example 3: %CSPROCES for Subsetting Holiday Data

This example processes data for non-holidays only. In this example, the shift CODE for holidays is 0, which is also the SAS IT Resource Management default holiday shift CODE.

```
%csproces('/raw/file',
          snmhpf snmiod,
          collectr=sunetmgr,
          subset=shift ne "0",
          _rc=retcode);
%put csproces return code= &retcode;
```

Example 4: %CSPROCES for HP-OVPA (formerly HP-PCS)

This example processes one day of global records from the HP-OVPA raw data files from host 1 and host 2 by using the *cspcsext* shell script. It routes the *stderr* file to *extract.msgs*.

```
x '/bin/rm ~/extract.msgs';
%csproces(pipe "cspcsext -xp d-1 -g host1 host2 2>> ~/extract.msgs",
          *,
          collectr=hp-ovpa,
          _rc=retcode);
%put csproces return code= &retcode;

   /* list extract messages in case an error occurred */
filename extrmsgs '~/extract.msgs';
data _null_;
   infile extrmsgs;
   input;
   put _infile_;
   run;
```

Note: The *cspcsext* shell script uses the PCS extract command to write PCS performance data to stdout. The script uses standard PCS options followed by a list of host names. Look for an example of the extract shell script in /misc/cspcsext. This file must be tailored for use at your site. △

Example 5: %CSPROCES HP Network or IBM Tivoli NetView

These examples process HP OpenView data. HP OpenView is used by both HP Network Node Manager and IBM Tivoli NetView for AIX to collect MIB data.

This example assumes that HP-OV is installed on the same machine on which you are running %CSPROCES.

```
%CSPROCES( '/usr/OV/databases/snmpCollect',
           hn2nix,
           collectr=HP-OV,
           _RC=retcode );
%put csproces return code is &retcode;
```

This example uses the RSHHOST= parameter to specify that HP-OV is installed on a different machine from the one on which you are running %CSPROCES. It specifies that the HP-OV machine is named 'netmon1' and that the command that is used to execute commands on the HP-OV machine is *remsh*. The raw data input directory that is used by %CSPROCES must be writable from both systems (for example, via NFS):

```
%CSPROCES( /nfs/mo/usr/OV/databases/snmpCollect,
           hn2nix hn2ift,
           rshhost=remsh netmon1,
           collectr=HP-OV,
           _RC=retcode );
%put csproces return code is &retcode;
```

This example assumes that you have created a sequential file of HP-OV data by using, for example, the script in **/misc/cpecoldm**. When you use %CSPROCES in this manner, it is generally less convenient than if you read the directory of files of raw HP-OV data directly. However, it might be necessary if you cannot easily obtain the required mount authorities to volumes on the HP-OV host at your site.

```
%CSPROCES( /usr/tmp/snmpCollect.01Jan96,
           hn2nix hn2ift,
           collectr=HP-OV,
           _rc=retcode );
%put csproces return code is &retcode;
```

Example 6: %CSPROCES for PROBE Agent Data

This example processes PROBE Agent data. Note that in Example 1 and Example 2, the name of the raw data file in the %CSPROCES step is enclosed in quotation marks, but the quotation marks are not required.

```
%csproces('/probex/logfile',
          prxbuf prxcld,
          collectr=probex,
          _rc=retcode);
%put csproces return code= &retcode;
```

In this example the incoming data is the same whether it is collected by SunNet Manager or PROBE Agent.

```
%csproces('/snm/logfile',
          spxbuf spxcld,
          collectr=sunetmgr,
          _rc=retcode);
%put csproces return code= &retcode;
```

Example 7: %CSPROCES for SPECTRUM Data Gateway

This example processes selected tables from a SPECTRUM Data Gateway data set. For additional information about processing SPECTRUM data, see "%CSCSIFMT" on page 202.

```
/*assign libref for SPECTRUM data */
libname specgway 'pathname/specgway';

/*create SAS formats in sitelib library */
%CSCSIFMT (modelds=specgway.model,
           _rc=retcode );
```

```
%put CSCSIFMT return code is &retcode;

/*process SPECTRUM data */
%csproces(specgway.stat,
          cciifp cxytse csibce csil23 csihi2 csipng cciigs,
          collectr=spectrum,
          _rc=retcode);
%put csproces return code is &retcode;
```

Example 8: %CSPROCES for SunNet Manager

This example processes a compressed SunNet Manager raw data file. It pipes the output of the *uncompress* command to the %CSPROCES macro. The example keeps data for only host 1 through host 5:

```
%csproces(pipe 'uncompress -c
/usr/snm/rawdata/log.15jan93.Z',
          snmhpf snmioc snmiod snmltp snmlup snmpgs snmrnc snmrns,
          collectr=sunetmgr,
          subset=machine in ('host1','host2','host3','host4','host5'),
          _rc=retcode);
%put csproces return code is &retcode;
```

%CSSNMSCH

Creates user-defined table and variable definitions for SunNet Manager data

%CSSNMSCH Overview

%CSSNMSCH builds %CPDDUTL control statements that are created from a SunNet Manager schema file. You can then use %CPDDUTL to apply the control statements to the active PDB. The result is one or more table definitions that correspond to the tables of metrics in the schema file.

%CSSNMSCH Syntax

%CSSNMSCH(
OUTFILE=*filename*
,**RPCIDS=***filename*
,**SCHEMA=***filename*
,**SNMNAMS=***filename*);

Details

OUTFILE=*filename*
identifies the path and name of the output file that is created by calling this macro. The file contains the %CPDDUTL control statements that define the PDB tables and variables, which correspond to the STAT data set or SPECTRUM data sets for %CSATR2DD and correspond to a SunNet Manager schema file for %CSSNMSCH.

RPCIDS=*filename*

specifies the path and name of the flat file that contains RPC ID definitions.

Note: SunNet Manager uses RPC IDs to map numbers to names (of tables), therefore determining how to log data. △

For more information, see the documentation from SunNet Manager.

SCHEMA=*filename*
specifies the path and name of the schema file to read.

SNMNAMS=*filename*
specifies the path and name of the flat file that contains SunNet Manager schema name definitions.

For more information, see the documentation from SunNet Manager.

%CSSNMSCH Notes

For further information on setting up your collector with SAS IT Resource Management, see the sections "General-Purpose Server Setup Documentation" and "Collector-Specific Setup Documentation" in the section "Locating Help" in "Chapter 1: Introduction to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

%CSSNMSCH Example

```
* Start SAS IT Resource Management and specify the active PDB.;

%CPSTART(  mode=batch,
           pdb=/u/testpdb,
           access=write  );

*  Create %CPDDUTL control statements from the schema file in /u/snm.ddutl;

%CSSNMSCH(  outfile=/u/snm.ddutl,
            rpcids=/misc/cpe/csrpc,
            schema=/usr/snm/schema/toaster.schema,
            snmnams=/u/nams );   /* Format like /misc/cpe/cssnmnam */

*  Create table and variable definitions from the %CPDDUTL control statements;

%CPDDUTL(  filename='/u/snm.ddutl',
           list=yes  );

*  Process the data that corresponds to a table in the new schema file, into the
*  table in the PDB;

%CSPROCES( /usr/snm/logfile,
           uutab1,
           collectr=SUNETMGR  );
```

%CWPROCES

Processes data into the detail level of the active PDB on Windows

%CWPROCES Overview

The %CWPROCES macro runs the process task, which includes reading and validating performance data and transforming the performance data from its original logged files into the detail level of the active performance database (PDB). In addition, %CWPROCES updates data dictionary entries that describe the amount of data in the PDB and describe the most recent date and time of processing data into the detail level of the tables.

%CWPROCES Syntax

%CWPROCES(
<*rawdata*>
,*tablist*
,**COLLECTR=***collector-name*
,**TOOLNM=***tool-name*
<,**AGELIMIT=***number*>
<,**COLDUMP=***command*>
<,**COLLVER=***version-number*>
<,**DELIM=***'delim-chars'*>
<,**EXITSRC=***exit-source*>
<,**GENLIB=***libref*>
<,**LOGFMT=**NORMAL | MULTITAB | SINGLETAB>
<,**MAXDATE=***date*>
<,**MINDATE=***date*>
<,**OPTIMIZE=**TIME | DISK>
<,**_RC=***macro-var-name*>
<,**SUBSET=***expression*>);

Details

rawdata

specifies the name of the raw data file. The name includes the complete directory path and name of the raw data file that is to be processed. This positional parameter is not required, but if it is specified, it must be first. If you omit this parameter, then you must use a comma as a placeholder. The name is the name of the raw file that contains data from your data collector (for example, HP Network Node Manager or IBM Tivoli NetView). The name of the file can be enclosed in matched single or double quotation marks, but quotation marks are not required.

Omit the *rawdata* parameter under the following circumstances:

- you want to process existing data in the detail level by using the AGELIMIT= or SUBSET= parameters. (For more information, see the AGELIMIT= parameter and the SUBSET= parameter.)
- you are processing data by using COLLECTR=GENERIC, in which case you would specify the GENLIB= parameter instead of *rawdata*.

tablist

lists one or more tables into which the logged data should be processed. *This positional parameter is required* and must be specified in the second position. If you specify more than one table, then use blanks to separate the tables in the list.

For HP OpenView Performance Agent input files, you can specify an asterisk (*) as the value of the tablist parameter if you want to process data from the raw data file into the appropriate tables.

COLLECTR=*collector-name*
specifies the name of the data collector or data source that provides the raw data. The value that you specify for the COLLECTR= parameter depends on which process macro you use. The following example uses %CSPROCES with the SUNETMGR collector:

```
%CSPROCES( ...collectr=sunetmgr,...);
```

Also, the TOOLNM= parameter may be required depending on which value you specify for the COLLECTR= parameter. The following example uses %CWPROCES with the WEBLOG collector:

```
%CWPROCES( ...collectr=weblog, toolnm=clf,...);
```

See the following COLLECTR - TOOLNM Parameter Values table for valid values of the COLLECTR= parameter and the TOOLNM= parameter. In the COLLECTR= Value and TOOLNM= Value columns of this table, the following process macro abbreviations apply:

%CM for %CMPROCES

%CP for %CPPROCES

%CS for %CSPROCES

%CW for %CWPROCES.

Table 2.12 COLLECTR - TOOLNM Parameter Values

COLLECTR= Value	Collector Description	TOOLNM= Value
GENERIC for %CM, %CP, %CS, and %CW	data collector or data source for which SAS IT Resource Management does not supply table definitions and the user has not installed table definitions	SASDS for %CM SASDS or CHARDELIM for %CP, %CS, and %CW
ACCUNX for %CP	UNIX accounting	SASDS
DCOLLECT for %CM	IBM DFSMS Data Collection Facility	MXG
EREP for %CM	IBM SYS1.LOGREC Error Recording	MXG
ETEWATCH for %CP	End-To-End Watch (transaction response measurement) by Candle	ETE12 (ETEWatch 1.2 from Candle) ETE13 (ETEWatch 1.3 from Candle) EBA (EBA logs from Candle)
HP-MWA for %CS and %CW	HP MeasureWare	MWA-BE or MWA-LE (See Note 6 following the table)

COLLECTR= Value	Collector Description	TOOLNM= Value
HP-OV for %CS and %CW	IBM Tivoli NetView or HP Network Node Manager	not applicable
HP-OV-C for %CS	IBM Tivoli NetView or HP Network Node Manager + Coerce	not applicable
HP-OVPA for %CS and %CW	HP OpenView Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
HP-PCS for %CS and %CW	HP Performance Collection Software	MWA-BE or MWA-LE (See Note 6 following the table)
HP-PCS for %CP	HP OpenView Reporter	SASACCESS
HP-VPPA for %CS and %CW	HP VantagePoint Performance Agent	MWA-BE or MWA-LE (See Note 6 following the table)
IMF for %CM	Boole & Babbage IMF from BMC	MXG
LSPW for %CS and %CW	Landmark Performance Works	CHARDELIM
NETCONV for %CP	Cisco IOS NetFlow	NETFLOW
NTSMF for %CM and %CP	Windows with Demand Technology's NTSMF product	MXG for %CM SASDS for %CP
PATROL for %CP	BMC Patrol	SASDS
PROBEX for %CS	Probe/X by Landmark in SunNetMgr format	not applicable
ROLMPBX for %CP	Rolm PBX	SASDS

COLLECTR= Value	Collector Description	TOOLNM= Value
SAPR3 for %CP	SAP R/3	SASADAPT (SAS IT Management Adapter for SAP) STATBIN (Statistics File) STATRFC (Using SAP R/3 API) (See Note 7 following the table)
SAR for %CP	UNIX sar command (system activity reporting)	SASDS
SITESCOP for %CP	SiteScope by Mercury Interactive	SASDS
SMF for %CM and %CP	IBM z/OS System Management Facilities	MXG
SPECTRUM for %CS	SPECTRUM by Aprisma	not applicable
SUNETMGR for %CS	SunNet Manager and Enterprise Manager	not applicable
TMON2CIC for %CM	Landmark “The Monitor for CICS/ESA 2”	MXG
TMONCIC8 for %CM	Landmark “The Monitor for CICS” (older)	MXG
TMONCICS for %CM	Landmark “The Monitor for CICS” V 1.3 and later	MXG
TMONDB2 for %CM	Landmark “The Monitor for DB2”	MXG
TMS for %CM	CA-TMS (Release 5 or later)	MXG
TPF for %CM	IBM Transaction Processing Facility	MXG

COLLECTR= Value	Collector Description	TOOLNM= Value
TRAKKER for %CS	Trakker by Concord Communications	not applicable
VISULIZR for %CP	Visualizer by BMC Software	SASACCESS
VMMON for %CM	VM Monitor	MXG
WEBLOG for %CP	Web server log	CLF (Common Log Format) ELF (Extended Log Format) IISORIG (Microsoft IIS original Weblog format) WEBETE (ETEWatch Web browser logs from Candle) WEBEBA (EBA logs from Candle)

1 *For COLLECTR=GENERIC:* For more information about using the Generic Collector Facility, see the chapter "Setup Case 3" in the *SAS IT Resource Management User's Guide* and the chapter "Setup Case 4" in the *SAS IT Resource Management User's Guide*, and the chapter "Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

   For more information about installing user-written or consultant-written table definitions, see "%CPPKGCOL" on page 144, "%CPRPTPKG" on page 177, and "%CPINSPKG" on page 124.

2 *For %CMPROCES:* The COLLECTR= parameter is required. The TOOLNM= parameter is also required.

3 *For %CPPROCES:* The COLLECTR= parameter is optional. If the parameter is not specified, then the default is COLLECTR=GENERIC. The TOOLNM= parameter is also optional. If the parameter is not specified, then the default is TOOLNM=SASDS.

4 *For %CSPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

5 *For %CWPROCES:* The COLLECTR= parameter is required. For information about the TOOLNM= parameter, see the Collector-Toolname Parameter Values table, above.

6 *For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA:* The TOOLNM= parameter is used only when you use a PIPE command for the raw data.

   For COLLECTR=HP-MWA or HP-VPPA or HP-OVPA, the TOOLNM= value is used to identify the endian type of the data. For example, data from Windows platforms is little-endian and data from most UNIX platforms is big-endian.

   Specify TOOLNM=MWA-BE if the incoming data is big-endian (BE). Specify TOOLNM=MWA-LE if the incoming data is little-endian (LE).

   If you use a PIPE command for the raw data and you do not specify the TOOLNM= parameter, SAS IT Resource Management attempts to determine the

endian type of the incoming data. If the attempt is successful, the processing step continues. If the attempt is not successful, the processing step terminates with a message.

Note: HP OpenView Performance Agent (HP-OVPA) is the latest name for these products: HP-PCS, HP-MWA, and HP-VPPA. △

7 *SAP:*

- *For SAP Release 4.6D and earlier, use TOOLNM=STATBIN or TOOLNM=STATRFC.*

  You must set one macro variable before you run %CPPROCES. Additionally, there are four other optional variables that you might need to set, depending on the data that you are processing.

  The required macro variable is

  - SAPVER - the SAP R/3 version.

  The optional macro variables are

  - SAPSYSNR - the SAP R/3 system number
  - SAPSYSNM - the SAP R/3 system name
  - SAPHOST - the SAP R/3 application server name
  - SAPPLAT - the platform on which the SAP STAT file was generated.

  You need to specify the SAPPLAT macro variable only if you are processing SAP STAT data that was generated in a Windows operating environment. The default value for the SAPPLAT macro variable enables you to read data that is generated in both UNIX and z/OS operating environments. However, if you read data that was generated in a Windows operating environment, then set SAPPLAT to a value of WIN. (For more information, see the examples for the %CPPROCES macro.)

  You must specify the SAPSYSNM, SAPHOST, and SAPSYSNR macro variables only if you use the SAP R/3 collector and only if you process a single data file.

  To populate these macro variables automatically, see the instructions in the RAWDATA= parameter for the %CPPROCES macro.

- *For SAP releases later than 4.6D, use TOOLNM=SASADAPT.*

  You can also use TOOLNM=SASADAPT with some earlier releases of SAP.

8 *If you are working with a Candle collector:*

- Use COLLECTR=ETEWATCH, TOOLNM=ETE12 or ETE13 or EBA when application and transaction analysis is desired.
- Use COLLECTR=WEBLOG, TOOLNM=WEBETE or WEBEBA when processing WebBrowser behavior module data or EBA data for which a hierarchical page analysis (for example, jobs —> Cary —> R&D) is desired.

9 *For COLLECTR=SMF on UNIX or Windows:* See “Example 8: Processing SMF data on UNIX or Windows” in the %CPPROCES Examples.

10 Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section “Collector Support Packages” in the chapter “Administration: Extensions to SAS IT Resource Management” in the *SAS IT Resource Management User’s Guide*.

TOOLNM=*tool-name*

typically specifies the name of the software that will be used to transform the raw data and to stage the data into SAS data sets or data set views. The value that is specified for this parameter is based on the value that you specify for the COLLECTR= parameter.

For more information about values of the TOOLNM= parameter, see the COLLECTR= parameter. The COLLECTR= parameter contains a table of COLLECTR= and TOOLNM= combinations and also contains collector-specific and macro-specific notes.

Note: Someone at your site may have written collector support for another data collector or data source, and packaged and installed the collector-support entities. If so, you have another collector-name tool-name combination for %CPPROCES that is not shown in this table. See the documentation for that collector support.

For more information, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*. △

AGELIMIT=*number*
is the number of days' worth of data to keep in the detail level of every table in the PDB. *The default is the value that is defined in the data dictionary for each table's detail level*. Use this parameter to override, on a specific run, the age limit that is specified in the data dictionary for a table in the PDB. When you specify this parameter, it does not reset the dictionary value.

If you do not specify input by using the *rawdata* parameter or the GENLIB= parameter, then this macro reprocesses any existing data in the detail level (and possibly ages out old data), based on the values of the AGELIMIT= and SUBSET= parameters.

Note: If AGELIMIT=0 at the detail level in the data dictionary, then your data is stored in the detail level until the next reduce or process task runs. Therefore, if the reduce task runs immediately after the process task, then the data is processed, reduced into the summary levels, and then removed from the detail level. However, if the process task runs again before the data has been reduced, then the existing data in the detail level is deleted before it is reduced to the summary levels. Using AGELIMIT=0 is useful if you are collecting and processing large amounts of data on a daily basis, but to avoid losing data you should always reduce the data immediately after it is processed. △

Note: For information about setting age limits for your data, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*. △

COLDUMP=*command*
specifies the name of the snmpColDump command to be used to dump HP-OV data at your site. *The default value for UNIX is /opt/OV/bin/snmpColDump and, by default, this command is installed in the directory where HP-OV is installed. The default for Windows environments is snmpColDump.*

For Windows environments, the value for this parameter is simply the snmpColDump command. You do not have to specify the full path for the location of the command. However, your path must contain the location of the snmpColDump utility.

COLLVER=*version-number*
specifies the version number of the SunNet Manager collector from which you are processing data. This parameter is used to process data from older versions of the collector. Specify this parameter only if the data was collected using SunNet Manager Version 1, and in this case use the value COLLVER=1. Otherwise, do not specify this parameter.

DELIM=*'delim_chars'*
specifies the character(s) that is used as a value separator in the character-delimited data. The delimiter character or list of delimiter characters must be enclosed in matched single or double quotation marks. Do not use a blank space to separate the items in the list. Specify a blank space in the list only if a blank space is one of the delimiters in your file.

If you do not specify a delimiter and you are processing character-delimited data, then the default value, a blank, is used.

Note: This parameter takes effect if TOOLNM=CHARDELIM or if COLLECTR=NTSMF.

EXITSRC=*exit-source*
specifies the location where exit routines (source programs) for SAS IT Resource Management are stored. The value of this parameter can be the name of a SAS catalog that is specified in the form of *libref.catname*, or it can be a specified location in your operating environment. For UNIX or Windows, this is the complete pathname of a directory. For z/OS, this is the fully qualified name of a partitioned data set.

If you specify a SAS catalog name, then use a SAS LIBNAME statement in order to associate the *libref* with the SAS library that contains the catalog.

If you want to process your data without using exit routines, then do not specify this parameter.

For the %CSPROCES and %CWPROCES macros, observations with duplicate values for the BY list variables are removed by default. If you want to prevent the removal of duplicate observations, use the PROC180 exit point that is described in "Exit Points for %CSPROCES and %CWPROCES" in the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

For more information on exit processing, see the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

Note: For the %CSPROCES and %CWPROCES macros, if you use process exits and you process HP OpenView Performance Agent data, then you must specify LOGFMT=SINGLETAB in order for your data to be processed. For more information, see the LOGFMT= parameter. △

GENLIB=*libref*
specifies the *libref* for the SAS library that contains the staged data that is to be processed into the PDB. The staged data has been created by user-provided software and is to be processed into a PDB by using COLLECTR=GENERIC. The data may be a SAS data set, a SAS DATA step view, a SAS SQL view, or a SAS/ACCESS view. For each table, the name of the data set, DATA step view, SQL view, or SAS/ACCESS view that has staged data for the table is specified by the table's external name parameter.

Use this parameter to specify the input data set or view for the SAS IT Resource Management process macro that you use to process your data, but only if you specify COLLECTR=GENERIC and TOOLNM=SASDS. If you specify this parameter for the %CPPROCES macro, then do not specify the RAWDATA= parameter. If you specify the GENLIB= parameter for the %CMPROCES macro or the %CSPROCES macro or the %CWPROCES macro, then do not specify the (positional) *rawdata* parameter.

LOGFMT=NORMAL | MULTITAB | SINGLETAB
indicates whether special handling is required to process the input data log with %CSPROCES or %CWPROCES. *The default value is MULTITAB. NORMAL is an*

alias for MULTITAB. This parameter is used only if COLLECTR=HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA.

If LOGFMT=MULTITAB or NORMAL and COLLECTR=HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA, then the SAS IT Resource Management process macro reads the input file that contains metrics from multiple tables or different metrics from the same table. The input file might contain data from multiple hosts that are concatenated together, but it must be a flat file (not data that is piped from the HP OpenView Performance Agent extract command). The data that is used with LOGFMT=MULTITAB or NORMAL must be clean data; that is, the data must not contain error or status messages between the records that are sometimes produced by extract when piping its output to a file.

If messages indicate that you need to specify a value with LOGFMT=MULTITAB, then use the format **LOGFMT=MULTITAB:nnnnn**, where **nnnn** is the maximum number of item IDs that can be processed. Normally the default value (10,000) will suffice.

If LOGFMT=SINGLETAB and COLLECTR=HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA, then the SAS IT Resource Management process macro requires the input data file to contain metrics from only one table (for example, GLOBAL, DISK, PROCESS, and so on) and precisely the same metrics must be collected from each occurrence of the table in the input file (data from different tables or different metrics from the same table are skipped and messages about the skip are written to the SAS log). However, the input file can contain data from multiple hosts that are concatenated together and can be either a flat file or piped output from the HP OpenView Performance Agent extract command.

Note: HP-OVPA is the latest name for the collectors HP-PCS, HP-MWA, and HP-VPPA. △

You must specify LOGFMT=SINGLETAB if

- you are using PIPE to read raw data with %CSPROCES
- you are processing HP-PCS, HP-MWA, HP-VPPA, or HP-OVPA data and you are using process exits (by using the EXITSRC= parameter).

MAXDATE=*date*

specifies the maximum acceptable date in incoming (new) data. The date value must use a valid SAS date-constant format. The format for valid SAS dates requires that the date be enclosed in quotation marks. For example,

```
'01-JAN-03'd
```

is an example of a valid SAS date for January 1, 2003, and

```
'01-JAN-2001'd
```

is an example of a valid SAS date for January 1, 2001. Do not use datetime constants such as

```
'01-JAN-02:23:30'dt
```

The following example shows how to use the MAXDATE= parameter to process data that has values for the DATETIME variable that are as far away as 10 days in the future. (Other parameters can be included as required.)

```
%CSPROCES(..., maxdate=TODAY()+10, ...) ;
```

The following example shows how to use the MAXDATE= parameter to process data up through the date of December 31, 2002. (Other parameters can be included as required.)

```
%CSPROCES(..., maxdate='31Dec2002'd, ...) ;
```

If the incoming data contains a DATETIME value that is greater than the value that you specify for this parameter, then processing stops before any new data is written to the detail level. *The default value is TODAY()+2*, which means that if any of the input data is dated more than 48 hours after the system date on the system where the data is being processed, then processing will stop.

MINDATE=*date*
specifies the minimum acceptable date in incoming (new) data. The date value must use a valid SAS date-constant format. The format for valid SAS dates requires that the date be enclosed in quotes. For example,

```
'01-JAN-02'd
```

is an example of a valid SAS date for January 1, 2002, and

```
'01-JAN-2001'd
```

is an example of a valid SAS date for January 1, 2001. Do not use datetime constants such as

```
'01-JAN-03:23:30'dt
```

If the incoming data contains a DATETIME value whose date is earlier than the value that you specify for this parameter, then processing stops. The default value is TODAY()-3650, which means that data up to 10 years old can be processed into the PDB.

OPTIMIZE=TIME | DISK
specifies whether time or disk space is more important to you when processing data. The default value is TIME. Do not specify this parameter if you specify COLLECTR=GENERIC.

When merging new data into the detail level of the PDB, SAS IT Resource Management creates a new data set, merges into it the incoming data and the data from the old data set, and then deletes the old data set. If the OPTIMIZE= parameter is set to (or defaults to) TIME, then the new data set is created on the same disk as the old data set. However, that disk may not have room to hold (temporarily) both the new and the old data sets. If that is the case, then specify OPTIMIZE=DISK, which causes the new data set to be created on the disk where the SAS WORK library resides and, at the end of the process step, to be copied to the disk where the old data set resided.

Note: When you specify OPTIMIZE=DISK, the copy of the table is stored in the library to which the libref WORK refers. If you do not want to use this library and disk, then use the **-work** SAS system option to change the WORK libref to refer to a different library. Be sure that the PDB and the WORK directory that you specify are on disks with sufficient space to store the largest detail table that you will process. △

If you are backloading data, then the space may need to be larger. For more details on the size of PDBs, refer to the "%CxPROCES Macros Recovery Procedures" in "Troubleshooting Batch Jobs That Fail" on page 655.

Note: %CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES. △

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

SUBSET=*expression*

specifies a data-subsetting expression or filtering criteria for new data that is being processed into the detail level and for existing data that is already stored in the detail level. You cannot use this parameter if you specify COLLECTR=GENERIC.

If you specify the *rawdata* parameter and if the expression is true, then the observation is stored in the detail level. If the expression is false, then the observation is not stored in or is deleted from (for currently stored data) the detail level. If you do not specify the *rawdata* parameter, then SUBSET= affects only the data that is currently stored in the detail level of the PDB.

The format of the value *expression* follows the rules for IF expressions. Refer to *SAS Language Reference* documentation for your current version of SAS to find details on specifying expressions.

%CWPROCES Notes

If you have made dictionary updates that require table views to be rebuilt, such as by adding variables, formula variables, or derived variables, then the views will be rebuilt automatically when %CWPROCES runs. For more information on updating tables in batch, see %CPDDUTL.

The MINDATE= and MAXDATE= parameters on %CWPROCES enable you to control whether future data is processed into the PDB. By default, these parameters prevent future data from being inadvertently processed into a PDB and thus from subsequently aging older data out of the PDB. However, for COLLECTR=GENERIC on %CWPROCES, this capability is implemented by using the CPFUTURE global macro variable instead of by using the MINDATE= and MAXDATE= parameters.

Note: By default, the %CWPROCES macro deletes duplicate observations—that is, those observations that have identical values in all of the BY or CLASS variables. If you want to prevent the deletion of duplicate observations, use the EXITSRC= parameter and the PROC180 exit. For more information, see the EXITSRC= parameter. △

%CWPROCES Example

Example 1: %CWPROCES for HP OpenView Performance Agent

This example processes data from a file that was created by using the HP OpenView Performance Agent (HP-OVPA) extract command under Windows.

```
%CWPROCES(e:\tmp\xfrdGLOBAL.bin,
          *,
          collectr=hp-ovpa,
```

```
          _rc=retcode);
%put CWPROCES return code=&retcode;
```

The next example processes one day of global records from a file that was created by using the HP OpenView Performance Agent (HP-OVPA) extract command on UNIX (the file was transmitted to Windows prior to processing, by using a transfer tool such as FTP with the binary option).

```
%CWPROCES(e:\tmp\globunix.bin,
          *,
          collectr=hp-ovpa,
          _rc=retcode);
%put CWPROCES return code=&retcode;
```

The next example shows how to use the MAXDATE= parameter to process data as far away as 10 days in the future. (Other parameters can be included as required.)

```
%CWPROCES(..., maxdate=TODAY()+10, ...)
```

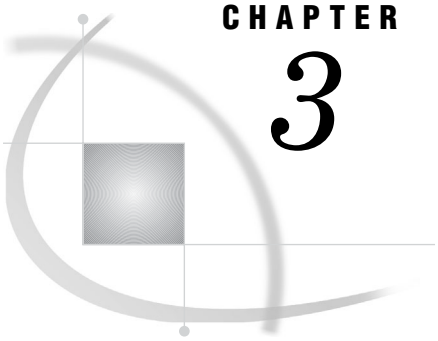
This example shows how to use the MAXDATE= parameter to process data up through the date of December 31, 2003. (Other parameters can be included as required.)

```
%CWPROCES(..., maxdate='31Dec2003'd, ...)
```

Example 2: %CWPROCES for HP Network Node Manager

This example processes HP OpenView data. HP OpenView is used by HP Network Node Manager to collect MIB data.

```
%CWPROCES( e:\OpenView\DataBases\snmpCollect,
           hn2nix,
           collectr=HP-OV,
           _rc=retcode );
%put cwproces return code is &retcode;
```



CHAPTER

3

Report Macros

Macros Used for Analyzing Data

You can use the following macros to design and generate reports on your data, as well as to create reports that you can display by using a Web browser. The report types include, but are not limited to, bar and pie charts, plots that have one or two Y axes, three-dimensional plots, spectrum plots, tabular reports, and Web versions of these reports.

- %CPCCHRT - charts multiple variables in one report (see "%CPCCHRT" on page 236)
- %CPCHART - charts values of a single variable (see "%CPCHART" on page 261)
- %CPEXCEPT - evaluates the exception rules in a folder (see "%CPEXCEPT" on page 291)
- %CPG3D - produces three-dimensional plots (see "%CPG3D" on page 294)
- %CPHTREE - provides organization and navigation for reports directed to the Web (see "%CPHTREE" on page 318)
- %CPIDTOPN - identifies the *N* top (or bottom) contributors to a phenomenon (see "%CPIDTOPN" on page 338)
- %CPMANRPT - deletes older reports in a SAS IT Resource Management Web gallery, based on specified criteria (see "%CPMANRPT" on page 356)
- %CPPLOT1 - plots one or more *Y* variables against an *X* variable by using one Y axis (see "%CPPLOT1" on page 362)

- %CPPLOT2 - plots two *Y* variables against an *X* variable by using two Y axes (see "%CPPLOT2" on page 389)
- %CPPRINT - prints a listing report (see "%CPPRINT" on page 414)
- %CPRUNRPT - runs one or more existing report definitions (see "%CPRUNRPT" on page 433)
- %CPSETHAX - creates a global macro variable that can be used to extend the horizontal axis on a report (see "%CPSETHAX" on page 450)
- %CPSPEC - generates a three-dimensional report with color as the third dimension (see "%CPSPEC" on page 453)
- %CPSRCRPT - runs existing SAS source code and provides access to global macro variables (see "%CPSRCRPT" on page 477)
- %CPTABRPT - produces tabular reports (see "%CPTABRPT" on page 507)
- %CPWEBINI - deletes all reports in a SAS IT Resource Management Web gallery (see "%CPWEBINI" on page 532)
- %CPXHTML - generates Web pages from exceptions that are found by %CPEXCEPT (see "%CPXHTML" on page 536)
- How the OUT*= Parameters Work Together (see "How the OUT*= Parameters Work Together" on page 551)
- Examples of the OUT*= Parameters (see "Examples of the OUT*= Parameters" on page 560)

Note: Within each report macro description, references to the reporting window interface or GUI typically relate to the reporting GUI for UNIX or PC environments. There might be some differences within the z/OS reporting GUI. △

%CPCCHRT

Generates charts for multiple analysis variables when each variable represents a category of data

%CPCCHRT Overview

The %CPCCHRT macro plots statistics for as many as 15 analysis variables (by using the *variable-label-pairs* parameter) in one chart. Each analysis variable represents a category or group of data and is represented by a separate bar, block, star peak, or pie slice. You can select any of the following types of charts:

- horizontal bar chart for both 2- and 3-dimensional views
- vertical bar chart for both 2- and 3-dimensional views
- pie chart for both 2- and 3-dimensional views
- block chart (BLOCK)
- star chart (STAR).

If the chart type is HBAR or HBAR3D, then the analysis variables and labels are displayed on the left (Y) axis. If the chart type is VBAR or VBAR3D, then the analysis variables and labels are displayed on the X axis. The other chart types do not use X and Y axes in order to display the analysis variables.

This type of report is useful when you have multiple analysis variables and each variable represents a category or group of data. For example, if you are analyzing packet rates, then your data might be displayed in groups, such as incoming packets,

local packets, and outgoing packets, which would be your analysis variables. All variables for the chart should have the same unit of measurement. By default, the mean value of each variable is charted. Use the STAT= parameter to change the statistic that is plotted.

For more information on the underlying procedure, refer to the SAS/GRAPH reference documentation for your current release.

%CPCCHRT Syntax

%CPCCHRT(
dataset
,*variable-label-pairs*
<,**ACROSS=***number*>
<,**AXIS=***value-list* | AXIS*n*>
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***BY-variable-list*>
<,**DOWN=***number*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**GROUP=***grouping-variable*>
<,**GRP_LAB=***label*> (obsolete; use LABELS=)
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**PATTRNID=**MIDPOINT | BY | GROUP>
<,**PERIOD=***time-interval*>
<,**PROCOPT=***string*>
<,**REDLVL=***PDB-level*>
<,**SGRP_LAB=***label*> (obsolete; use LABELS=)
<,**SMALLDEV=***device-driver*>
<,**STACKVAR=***variable*>
<,**STAT=***statistic*>
<,**STATLAB=***'label'*>
<,**STMTOPT=***string*>
<,**SUBGROUP=***variable*>
<,**TABTYPE=**INTERVAL | EVENT>
<,**TYPE=***chart-type*>
<,**VREF=***value-list*>
<,**WEBSTYLE=***style*>
<,**WEIGHT=***weight-variable*>
<,**WHERE=***where-expression*>);

Details

dataset
specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

variable-label-pairs
specifies a list of pairs. (For %CPCHART, you can use only one pair.) Each pair consists of the name of one analysis variable and the label for that analysis variable. Each label has a maximum length of 16 characters.

The maximum number of pairs that you can specify is 15, depending on the type of report that you are running. *For a detailed explanation of the variables that are displayed on each axis and the maximum number of variables that can be specified for the report, refer to the "Overview" section of the macro description.*

If you specify multiple analysis variables, then the unit of measure for all of the analysis variables must be the same and the range of values should be similar.

The analysis variable is typically displayed on the left (Y) axis. However, certain combinations of parameters might affect the axis on which the analysis variable is displayed or the number of variables that can be specified, depending on the report macro that you use.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of a label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in a label.

You are required to specify at least one variable in this positional parameter. However, you are not required to specify labels when you use this parameter. In fact, using the LABELS= parameter to specify the labels is preferred. Because this is a positional parameter, you must use a comma as a placeholder for the label when you specify more than one variable but do not specify labels. For example, to specify three variables and no labels, you could specify **`var1,,var2,,var3,KEYWORD=...`**. In this example, you have specified variable 1 but no label for variable 1, variable 2 but no label for variable 2, and variable 3 but no label for variable 3. You then specify any keyword parameters, such as the LABELS= parameter, that you want to use in the report definition.

You can specify the labels by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the LABELS= parameter,

then any labels that are specified by using this parameter are ignored. If labels are not specified by using this parameter and not specified by using the LABELS= parameter, then the variables' labels in the data dictionary are used.

ACROSS=*number*
specifies the number of star or pie charts to place across a single graph page. *The default is 1.*

AXIS=*value-list* | AXIS*n*
specifies the values for the major tick marks on the analysis axis (the X axis for horizontal bar charts, and the Y axis for vertical bar charts) or an axis number that has been previously defined in a palette. You can specify the value in one of two ways:

- AXIS*n*, where *n* represents an axis number that you have already defined in the palette that you are currently using for this report.

  If you specify an axis number, then you should also specify the PALETTE= parameter to indicate the name of the palette that contains this axis, or you should define the axis yourself before calling the macro. For more information about defining an axis, see the SAS/GRAPH documentation for your current release of SAS.

- a list of values for the axis.

These examples illustrate ways that you can specify the list of values (*value-list*):

```
n n . . . n

or

n TO n
   <BY increment>
```

You can also combine these methods as shown below:

```
n n . . . n TO n
   <BY increment> n . . . n
```

You can separate the items in the list with spaces or commas.

For more information about the *value-list*, see the ORDER= parameter in the "AXIS Statement" section of the SAS/GRAPH documentation for your current version of SAS.

Note: The values for AXIS= must be specified as actual values, not as formatted values. Thus, the tick marks for time values must be either a count of seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **dd:mmm:yyyy:hh:mm:ss**, where **dd** is day, **mmm** is month, **yyyy** is year (in two-digit or four-digit notation), **hh** is hour (using a 24-hour clock), **mm** is minute, and **ss** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*BY-variable-list*
lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

DOWN=*number*
specifies the number of star or pie charts to place vertically on a single graph page. *The default is 1.*

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*
lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

GROUP=*grouping-variable*
is the name of a variable that subsets the observations. Graphic elements (bars, blocks, slices of pie, and peaks of stars), which represent analysis variables, are

displayed in groups. There is one group for each value of the group variable. If the TYPE= parameter for this macro is HBAR, HBAR3D, VBAR, VBAR3D, or BLOCK, then the groups are displayed in the form of a chart, with all groups on one chart. If TYPE= STAR or PIE, the analysis variables are represented by peaks of stars or slices of pie and a separate chart is displayed for each unique value of the group variable. In the %CPCCHRT macro, you can use the GROUP= parameter with either the STACKVAR= parameter or the SUBGROUP= parameter, but not both.

GRP_LAB=*label (obsolete; use LABELS=)*
specifies the label for the variable that is specified by the GROUP= parameter. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify a label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

HTMLDIR=*directory-name | PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhdir=c:\www\hreports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report

macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △

- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*

specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhurl=http://www.reporter.com/reports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*

specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report

macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △

- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)

specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).
- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

```
options
  appletloc=
```

```
"http://path-to-copy-of-graphapp-jarfile";
```

where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*

specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

OUTDESC=*output-description*

if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*

for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT*: WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points

to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename* | *entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)

- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT:* This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

PALETTE=*palette-name*

specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **`sasuser.palette.win`** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path

from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PATTRNID=MIDPOINT | BY | GROUP

specifies when to use the next color or pattern in the palette that is specified in the PALETTE= parameter. MIDPOINT indicates that the colors or patterns on the bars or blocks in the chart will be different for each value of your axis variable. BY indicates that the colors or patterns on the bars or blocks in the chart will change as the BY variable values change. That is, all bars or blocks in the same BY group have the same color. GROUP indicates that the colors or patterns on the bars or blocks in the chart will change with the value of the group variable. *The default value is MIDPOINT.*

This parameter should be used only for charts with TYPE= HBAR, HBAR3D, VBAR, VBAR3D, or BLOCK.

This parameter is not used for TYPE=PIE or TYPE=STAR. If you specify this parameter for those report types, then the PATTRNID= parameter is ignored.

You can specify PATTRNID= if you specify the STACK= parameter with only one analysis variable.

The PATTRNID= parameter should not be used and is ignored if you have specified the SUBGROUP= or STACK= parameter with multiple analysis variables. In the case of subdivisions (by use of the SUBGROUP= or STACK= parameter), each subdivision automatically has its own color or pattern.

PERIOD=*time-interval*

is the summarization period for the report. *The default is ASIS*. For values other than ASIS, the values for the time period within a class or category are combined (as specified in the STAT= parameter) to form a single point on a plot, a bar on a chart, or a row or column in a table.

The following list provides the period name, the effect that it has on the report, and an example of how to specify the period.

- ASIS - leaves datetime in its present form. Example: 09Jun2003:14:37:25
- 15MIN - transforms the time to the first minute of the 15-minute period in the hour and retains the date. Example: 09Jun2003:14:30
- HOUR - transforms the time to the first minute of the hour and retains the date. Example: 09Jun2003:14:00
- 24HOUR - transforms the time to its hour component and omits the date. Range: 0–23. Example: 14
- DATE - omits the time and retains the date. Example: 09Jun2003
- WHOLEDAY - obsolete; use DATE.
- WEEKDAY - transforms the day to the day of the week (where 1=Sunday, 2=Monday, and so on) and omits the month, year, and time. Range: 1–7, which displays as Sunday through Saturday. Example: Monday
- MONTHDAY - omits everything but the day of the month. Range: 1–31. Example: 9
- YEARDAY - transforms the day to the day of the year and omits the month, year, and time. Range: 1–366. Example: 160
- WEEK - transforms the date to the first day of the week as defined by Start-of-Week and omits the time. (Start-of-Week is a PDB property that you can specify with the %CPPDBOPT macro. By default, Start-of-Week is Sunday.) Example: 08Jun2003
- MONTH - omits the day and the time. Example: Jun2003
- YEARMONTH - omits everything but the month of the year. Range: 1–12. Example: 6
- QTR - transforms the month to the first month of the quarter and omits the day and time. Example: Apr2003

Note: You can change the width of the bar that appears on the 3-dimensional plot generated by %CPSPEC. If you specify PERIOD= ASIS, or 15MIN, or HOUR, the bar will be thinner in order to accommodate the increased number of points per line that is typical with shorter periods. △

PROCOPT=*string*

where *string* is any text that is permitted on the PROC statement of the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro. This allows expert SAS/GRAPH users to take advantage of options that are not supported by SAS IT Resource Management macros or new options for which SAS IT Resource Management has not yet added support.

For %CPPLOT1, %CPPLOT2, and %CPSPEC, the corresponding procedure in SAS/GRAPH is PROC GPLOT.

For %CPCCHRT and CPCHART, the corresponding procedure in SAS/GRAPH is PROC GCHART.

For %CPG3D, the corresponding procedure in SAS/GRAPH is PROC G3D.

For %CPTABRPT, the corresponding procedure in the SAS System is PROC TABULATE.

For example, you might specify an annotate data set:

```
%CPPLOT1(....,
         PROCOPT=ANNO=WORK.MYANNO,
         ....);
```

The default value is blank (no value).

For more details about what options are valid on the PROC GPLOT, PROC G3D, and PROC GCHART statements, see your SAS/GRAPH documentation. For information about PROC TABULATE, see your SAS System documentation.

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

SGRP_LAB=*label (obsolete; use LABELS=)*
specifies the label for the variable that is specified by the SUBGROUP= parameter. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify a label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC*.

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

STACKVAR=*variable*
specifies the name of a variable by which your data is subdivided. A separate bar or block is produced for each value of the STACKVAR variable. Each bar is divided into sections, one for each analysis variable. The variable contains the same information as the SUBGROUP= parameter, but the presentation is different. Do not use the STACKVAR= parameter if you use the SUBGROUP= parameter.

Note: You cannot use STACKVAR= with TYPE=PIE or with TYPE=STAR. △

STAT=*statistic*

specifies what statistic to use to summarize data for the analysis variable(s) when the analysis variable (*variable-label-pairs*) is graphed and when the PERIOD= parameter is not ASIS. *The default statistic is MEAN*. Valid values are as follows:

MEAN
is the arithmetic average. Mean is one measure that is used to describe the center of a distribution of values. Other measures include mode and median.

SUM
is the sum of values for all observations.

STATLAB=*'label'*
specifies how to label the response axis if you do not want to use the name of the statistic. If you specify this parameter, then you should specify a label for the statistic that is specified in the STAT= parameter on this macro.

Note: You can specify only one label for the statistic. △
To use a label such as Monthly Total, you could specify the following:

```
STAT= sum,
STATLAB=Monthly Total
```

Do not include commas, equal signs, parentheses, or ampersands in the labels.

STMTOPT=*string*
where *string* is any text that is permitted as an option for the statement in the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro.

For %CPPLOT1 and %CPPLOT2, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPCCHRT and %CPCHART, the statement corresponds to the value of the TYPE parameter. For example, if TYPE=HBAR, then the statement will be the HBAR statement in PROC GCHART, and *string* will be added after the *'/'* in that statement.

For %CPSPEC, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPG3D, the corresponding statement in SAS/GRAPH is the SCATTER statement in PROC G3D.

For %CPTABRPT, the corresponding statement in SAS is the TABLE statement in PROC TABULATE.

If you want to customize the labeling and details of the axis for the group variable, you can specify an axis statement such as AXIS3 and then instruct %CPCHART to use it:

```
AXIS3 LABEL=(COLOR=BLUE);
%CPCHART(table,
         ....
         GROUP=gvar,
         STMTOPT=GAXIS=AXIS3,
         ....);
```

The default value is blank (no value).

For more details about what options are available for these statements, see the information about PROC GCHART, PROC G3D, and PROC GPLOT in your SAS/GRAPH documentation and PROC TABULATE in your SAS System documentation.

SUBGROUP=*variable*
specifies the name of a variable. This variable subdivides the groups that are specified by the value of the GROUP= parameter. Each subdivision becomes a

section of a bar or block. The variable contains the same information as the STACKVAR= parameter, but the presentation is different. (In the %CPCCHRT macro, you can use the STACKVAR= parameter or the SUBGROUP= parameter, or neither, but not both.)

You cannot specify a SUBGROUP= variable if you specify TYPE= PIE or TYPE=STAR.

TABTYPE=INTERVAL | EVENT
specifies whether each observation in the data that is read into the table represents a specified interval of time or whether each observation was logged in response to an event, such as when a job began or ended. *If the data is in a view or data set in the detail, day, week, month, or year level of the active PDB, then the default is the value that is specified in the table's definition in the active PDB's data dictionary.*

INTERVAL
each observation represents an interval of time

EVENT
each observation represents an event.

TYPE=*chart-type*
is the type of chart to produce. *The default type is HBAR.* Valid values are as follows:

BLOCK
creates a three-dimensional block chart in which the height of each block represents the statistic for the analysis variable that is being charted.

HBAR
creates a horizontal bar chart in which the length of each bar represents the statistic for the analysis variable that is being charted.

HBAR3D
creates a horizontal bar chart like HBAR, but displays the 2-dimensional results in a way that simulates a 3-dimensional view.

PIE
creates a circular pie chart in which the size of each slice represents the statistic for the analysis variable that is being charted.

PIE3D
creates a circular pie chart like PIE, but displays the 2-dimensional results in a way that simulates a 3-dimensional view.

STAR
creates a star chart in which the size of each peak represents the statistic for the analysis variable that is being charted.

VBAR
creates a vertical bar chart in which the height of each bar represents the statistic for the analysis variable that is being charted.

VBAR3D
creates a vertical bar chart like VBAR, but displays the 2-dimensional results in a way that simulates a 3-dimensional view.

Note: If the value of the analysis variable is zero or missing for every slice (in a TYPE=PIE chart) or every peak (in a TYPE=STAR chart), then the report definition stops. You can use the WHERE= parameter to circumvent this, by specifying WHERE= *analysis-variable*>0. The values that are eliminated from pies and stars are ones that would not have been in the pie or star in any case. △

VREF=*value-list*
indicates values on the response axis where you want to draw one or more lines to use as reference lines. Separate the items in the *value-list* with one or more blanks.

value-list can also be in the form of:

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

The separator can be either a comma or blanks.

Note: This parameter is for horizontal bar charts, vertical bar charts, and block charts only. This parameter is ignored by pie charts and star charts. △

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WEIGHT=*weight-variable*
specifies the alternate variable by which to weight (multiply) statistical calculations. If no WEIGHT= variable is specified and the table type is INTERVAL, then the variable DURATION is used as the weight.

For example, if most observations have a value of 15 for the weight variable and one observation has a value of 30 for the weight variable, then that observation has twice the weight of each of the other observations in any calculations.

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

%CPCCHRT Notes

There are several ways to create charts in order to analyze your data. The following list describes the different styles of charts that you can produce by using this macro.

- %CPCCHRT Style 1
  - This style produces one pie or star or one set of bars or blocks.
  - In the pie or star, there is one slice or peak for each analysis variable. In the set of bars or blocks, there is one bar or block for each analysis variable.
  - The size of a pie slice or star peak or the length of a bar or block represents the specified statistic of the analysis variable.
  - You must specify at least two analysis variables; you can specify as many as 15 analysis variables.
- %CPCCHRT Style 2
  - This style produces, for each value of the group variable, one pie/star or one set of bars/blocks.
  - In the pie or star, there is one slice or peak for each analysis variable. In the set of bars or blocks, there is one bar or block for each analysis variable.
  - The size of a pie slice or star peak or the length of a bar or block represents the specified statistic of the analysis variable.
  - You must specify at least two analysis variables; you can specify as many as 15 analysis variables. You must specify one group variable.

- %CPCCHRT Style 3
  - This style produces, for each value of the group variable, one set of bars or blocks.
  - In a set, the analysis variable is represented by a bar or block. Within a bar or block, there are subdivisions. Each subdivision represents one value of the subgroup variable.
  - The length of a bar or block represents the specified statistic of the analysis variable.
  - You must specify at least one analysis variable; you can specify as many as 15 analysis variables. You must specify one group variable and one subgroup variable.
- %CPCCHRT Style 4
  - This style produces, for each value of the group variable, one set of bars or blocks.
  - In a set, a stack variable is represented by a bar or block. Within a bar or block, there are subdivisions if there is more than one analysis variable. Each subdivision represents an analysis variable.
  - The length of a bar or block represents the sum of the specified statistic of the analysis variables.
  - You must specify at least one analysis variable; you can specify as many as 15 analysis variables. You must specify one group variable and one stack variable.
- For all styles you can specify one optional statistic and one optional weight variable. The statistic is used to collapse (summarize and replace) data if the Summary Time Period is not AS IS. The default statistic is SUM for star or pie charts and MEAN for bar and block charts.

  In a table of type interval, if no alternate weight variable is specified, then DURATION is used as the weight variable. (For each observation, the value of the weight variable is used to weight the values of the analysis variables during report-time summarization, such as the summarization when Summary Time Period is not AS IS.)
- For these styles you can also specify *n* optional variables by using the BY= parameter. No BY variables are required. If there is one BY variable, then a separate section of the report is generated for each value of the BY variable. If there is more than one BY variable, then a separate section of the report is generated for each unique combination of the values of the BY variables.
- The variables that you specify in the *classname*, *variable-label-pairs*, BY=, FORMATS=, GROUP=, LABELS=, SUBGROUP=, and WEIGHT= parameters must exist in the table that is specified by the *dataset* parameter and in the level that is specified by the *dataset* or REDLVL= parameter. If the level is detail, then the variable must have a KEPT status of YES. If the level is day, week, month, or year, then the variable must be in the level's class variables list or ID variables list or must be a statistic that has been selected for calculation.

Refer to "%CPCHART" on page 261 for additional chart styles.

Note: Both the STACKVAR= and SUBGROUP= parameters subdivide the data; they cannot be used at the same time. The difference between them is how the subdivisions are presented. The STACKVAR= parameter uses separate bars for the subdivisions. The SUBGROUP= parameter uses separate sections of the bar for the subdivisions. △

%CPCCHRT Example

This example generates a report with horizontal bars, one bar for each value (0-9) of the performance group variable. Each bar is subdivided into segments, one segment for

each value of the shift variable. The length of each bar and each subdivision of a bar is based on the value of the ACFRMTN variable (the number of active frames) that are in the observations with the specified performance group variable and the specified shift variable. The report's destination is an entry in a SAS catalog that is named SASUSER.CPUPGMS.CCHRTEX1.GRSEG. Specify a catalog or filename appropriate for your operating environment.

```
options pagesize=60 linesize=132;

goptions reset=title reset=footnote colors=(w r b g);
%cpcchrt(tababc,acfrmtn,Active frame
         ,group=perfgrp
         ,outmode=catalog
         ,subgroup=shift
         ,type=hbar
         ,weight=duration
         ,yval=16
         ,redlvl=detail
         ,where=perfgrp le 9
         ,outloc=sasuser.cpupgms
         ,outname=cchrtex1);
```

%CPCHART

Generates single-variable charts

%CPCHART Overview

The %CPCHART macro charts the values of a single analysis variable (*variable-label-pair*) that is categorized by the values of a single class variable (*classname*). This means that you can specify one analysis variable and label by using the *variable-label-pair* parameter and one class variable and label by using the *classname* parameter and the *classlabel* parameter. This macro can produce any of these types of charts:

- horizontal bar chart for both 2- and 3-dimensional views
- vertical bar chart for both 2- and 3-dimensional views
- pie chart for both 2- and 3-dimensional views
- block chart
- star chart.

Each bar, pie slice, block, or star peak represents a statistic based on the analysis variable for a discrete value of the classification variable. Use the STAT= parameter to define the type of statistic calculated for the analysis variable. A separate bar, pie slice, block, or star peak is produced for each value of the class variable.

For more information on the underlying procedure, refer to the GCHART procedure in the SAS/GRAPH documentation for your current version of SAS.

%CPCHART Syntax

%CPCHART(
dataset

,*classname*
,<*classlabel*> (obsolete; use LABELS=)
,*variable-label-pairs*
<,**ACROSS=***number*>
<,**AXIS=***value-list* | AXIS*n*>
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***BY-variable-list*>
<,**DOWN=***number*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**GROUP=***grouping-variable*>
<,**GRP_LAB=***label*> (obsolete; use LABELS=)
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**ORDER=**DESCENDING | ASCENDING | MIDPOINT>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**PATTRNID=**MIDPOINT | BY | GROUP>
<,**PERIOD=***time-interval*>
<,**PROCOPT=***string*>
<,**REDLVL=***PDB-level*>
<,**SGRP_LAB=***label*>(obsolete; use LABELS=)
<,**SMALLDEV=***device-driver*>
<,**STAT=***statistic*>
<,**STATLAB=***'label'*>
<,**STMTOPT=***string*>
<,**SUBGROUP=***variable*>
<,**TABTYPE=**INTERVAL | EVENT>
<,**TOPN=***number*>
<,**TYPE=***chart-type*>
<,**VREF=***value-list*>
<,**WEBSTYLE=***style*>
<,**WEIGHT=***weight-variable*>
<,**WHERE=***where-expression*>
<,**YVAL=***number*>
<,**ZERO=**YES | NO >);

Details

dataset
specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

classname
specifies the name of the class variable for the report. For each unique value of the class variable, the macro creates a separate portion of the chart, such as a slice of pie on a pie chart. All values of the class variable are included in a single chart. This is different from the BY= parameter, which creates separate charts for each unique value of the BY= parameter. *This positional parameter is required*.

classlabel
specifies the label for the class variable. The label has a maximum length of 16 characters.

This positional parameter is not required. However, if you do not specify this parameter, then you must specify a comma as a placeholder.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify a label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

variable-label-pairs
specifies a list of pairs. (For %CPCHART, you can use only one pair.) Each pair consists of the name of one analysis variable and the label for that analysis variable. Each label has a maximum length of 16 characters.

The maximum number of pairs that you can specify is 15, depending on the type of report that you are running. *For a detailed explanation of the variables that are displayed on each axis and the maximum number of variables that can be specified for the report, refer to the "Overview" section of the macro description.*

If you specify multiple analysis variables, then the unit of measure for all of the analysis variables must be the same and the range of values should be similar.

The analysis variable is typically displayed on the left (Y) axis. However, certain combinations of parameters might affect the axis on which the analysis variable is displayed or the number of variables that can be specified, depending on the report macro that you use.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or in double quotation marks, but the

quotation marks are not required. If the body of a label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in a label.

You are required to specify at least one variable in this positional parameter. However, you are not required to specify labels when you use this parameter. In fact, using the LABELS= parameter to specify the labels is preferred. Because this is a positional parameter, you must use a comma as a placeholder for the label when you specify more than one variable but do not specify labels. For example, to specify three variables and no labels, you could specify **`var1,,var2,,var3,KEYWORD=...`**. In this example, you have specified variable 1 but no label for variable 1, variable 2 but no label for variable 2, and variable 3 but no label for variable 3. You then specify any keyword parameters, such as the LABELS= parameter, that you want to use in the report definition.

You can specify the labels by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the LABELS= parameter, then any labels that are specified by using this parameter are ignored. If labels are not specified by using this parameter and not specified by using the LABELS= parameter, then the variables' labels in the data dictionary are used.

ACROSS=*number*
specifies the number of star or pie charts to place across a single graph page. *The default is 1.*

AXIS=*value-list* | AXIS*n*
specifies the values for the major tick marks on the analysis axis (the X axis for horizontal bar charts, and the Y axis for vertical bar charts) or an axis number that has been previously defined in a palette. You can specify the value in one of two ways:

- AXIS*n*, where *n* represents an axis number that you have already defined in the palette that you are currently using for this report.

  If you specify an axis number, then you should also specify the PALETTE= parameter to indicate the name of the palette that contains this axis, or you should define the axis yourself before calling the macro. For more information about defining an axis, see the SAS/GRAPH documentation for your current release of SAS.
- a list of values for the axis.

These examples illustrate ways that you can specify the list of values (*value-list*):

```
n n . . . n

or

n TO n
   <BY increment>
```

You can also combine these methods as shown below:

```
n n . . . n TO n
   <BY increment> n . . . n
```

You can separate the items in the list with spaces or commas.

For more information about the *value-list*, see the ORDER= parameter in the "AXIS Statement" section of the SAS/GRAPH documentation for your current version of SAS.

Note: The values for AXIS= must be specified as actual values, not as formatted values. Thus, the tick marks for time values must be either a count of

seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **dd:mmm:yyyy:hh:mm:ss**, where **dd** is day, **mmm** is month, **yyyy** is year (in two-digit or four-digit notation), **hh** is hour (using a 24-hour clock), **mm** is minute, and **ss** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*BY-variable-list*
lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

DOWN=*number*
specifies the number of star or pie charts to place vertically on a single graph page. *The default is 1.*

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*
lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least

one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

GROUP=*grouping-variable*
is the name of a variable that subsets the observations. Graphic elements (bars, blocks, slices of pie, and peaks of stars), which represent analysis variables, are displayed in groups. There is one group for each value of the group variable. If the TYPE= parameter for this macro is HBAR, HBAR3D, VBAR, VBAR3D, or BLOCK, then the groups are displayed in the form of a chart, with all groups on one chart. If TYPE= STAR or PIE, the analysis variables are represented by peaks of stars or slices of pie and a separate chart is displayed for each unique value of the group variable. In the %CPCCHRT macro, you can use the GROUP= parameter with either the STACKVAR= parameter or the SUBGROUP= parameter, but not both.

GRP_LAB=*label (obsolete; use LABELS=)*
specifies the label for the variable that is specified by the GROUP= parameter. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify a label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the

macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is,

HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*

specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the

macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **welcome.htm** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the

value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myiurl=http://www.reporter.com/reports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)

specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).

- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)

- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The

location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

  Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **`View Source`**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*

specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **`hh:mm`**, where **`hh`** is hour (using a 24-hour clock) and **`mm`** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **`00:00`**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **`'12Apr2003:01:30'dt`**, then **`11Apr2003`** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **`11Apr2003`**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **`'12Apr2003:04:31'dt`**, then **`12Apr2003`** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **`12Apr2003`**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

ORDER=DESCENDING | ASCENDING | MIDPOINT

indicates the order in which the graphic elements (bars, blocks, pie slices, and star peaks) are to be placed in the chart.

If the TOPN= parameter is specified, then the valid values are ASCENDING and DESCENDING, and DESCENDING is the default. If the TOPN= parameter is not specified, then the valid value is MIDPOINT, and MIDPOINT is the default.

These values are defined as follows:

DESCENDING
arranges in order from the highest value of the analysis statistic to the lowest. If the TOPN= parameter is specified, then the chart becomes a "TopN" chart.

ASCENDING
arranges in order from the lowest value of the analysis statistic to the highest. If the TOPN= parameter is specified, then the chart becomes a "BottomN" chart.

MIDPOINT
arranges in order alphanumerically, based on the value of the chart variable. The TOPN= parameter must not be specified if you specify ORDER=MIDPOINT.

For example, if the class variable is MACHINE, the type of the chart is bar, and the analysis statistics (and thus the bar lengths) are 48 for the bar for MACHINE1, 12 for the bar for MACHINE2, and 24 for the bar for MACHINE3, then the results are as follows:

- ORDER=DESCENDING results in the order MACHINE1 MACHINE3 MACHINE2
- ORDER=ASCENDING results in the order MACHINE2 MACHINE3 MACHINE1
- ORDER=MIDPOINT results in the order MACHINE1 MACHINE2 MACHINE3.

For more information, see the Chart Variables and Midpoint Selection topic in the GCHART procedure, in the SAS/GRAPH documentation for your current release of SAS.

OUTDESC=*output-description*
if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **`welcome.htm`** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is

stored with the report in the catalog that is specified in the OUTLOC= parameter.
If OUTMODE=LPCAT or OUTMODE=GRAPHCAT,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*
for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT:* WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT:* This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename* | *entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT*: G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)

- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

PALETTE=*palette-name*

specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **`sasuser.palette.win`** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/

Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PATTRNID=MIDPOINT | BY | GROUP

specifies when to use the next color or pattern in the palette that is specified in the PALETTE= parameter. MIDPOINT indicates that the colors or patterns on the bars or blocks in the chart will be different for each value of your axis variable. BY indicates that the colors or patterns on the bars or blocks in the chart will change as the BY variable values change. That is, all bars or blocks in the same BY group have the same color. GROUP indicates that the colors or patterns on the bars or blocks in the chart will change with the value of the group variable. *The default value is MIDPOINT.*

This parameter should be used only for charts with TYPE= HBAR, HBAR3D, VBAR, VBAR3D, or BLOCK.

This parameter is not used for TYPE=PIE or TYPE=STAR. If you specify this parameter for those report types, then the PATTRNID= parameter is ignored.

You can specify PATTRNID= if you specify the STACK= parameter with only one analysis variable.

The PATTRNID= parameter should not be used and is ignored if you have specified the SUBGROUP= or STACK= parameter with multiple analysis variables. In the case of subdivisions (by use of the SUBGROUP= or STACK= parameter), each subdivision automatically has its own color or pattern.

PERIOD=*time-interval*

is the summarization period for the report. *The default is ASIS*. For values other than ASIS, the values for the time period within a class or category are combined (as specified in the STAT= parameter) to form a single point on a plot, a bar on a chart, or a row or column in a table.

The following list provides the period name, the effect that it has on the report, and an example of how to specify the period.

- ASIS - leaves datetime in its present form. Example: 09Jun2003:14:37:25
- 15MIN - transforms the time to the first minute of the 15-minute period in the hour and retains the date. Example: 09Jun2003:14:30

- HOUR - transforms the time to the first minute of the hour and retains the date. Example: 09Jun2003:14:00
- 24HOUR - transforms the time to its hour component and omits the date. Range: 0–23. Example: 14
- DATE - omits the time and retains the date. Example: 09Jun2003
- WHOLEDAY - obsolete; use DATE.
- WEEKDAY - transforms the day to the day of the week (where 1=Sunday, 2=Monday, and so on) and omits the month, year, and time. Range: 1–7, which displays as Sunday through Saturday. Example: Monday
- MONTHDAY - omits everything but the day of the month. Range: 1–31. Example: 9
- YEARDAY - transforms the day to the day of the year and omits the month, year, and time. Range: 1–366. Example: 160
- WEEK - transforms the date to the first day of the week as defined by Start-of-Week and omits the time. (Start-of-Week is a PDB property that you can specify with the %CPPDBOPT macro. By default, Start-of-Week is Sunday.) Example: 08Jun2003
- MONTH - omits the day and the time. Example: Jun2003
- YEARMONTH - omits everything but the month of the year. Range: 1–12. Example: 6
- QTR - transforms the month to the first month of the quarter and omits the day and time. Example: Apr2003

Note: You can change the width of the bar that appears on the 3-dimensional plot generated by %CPSPEC. If you specify PERIOD= ASIS, or 15MIN, or HOUR, the bar will be thinner in order to accommodate the increased number of points per line that is typical with shorter periods. △

PROCOPT=*string*

where *string* is any text that is permitted on the PROC statement of the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro. This allows expert SAS/GRAPH users to take advantage of options that are not supported by SAS IT Resource Management macros or new options for which SAS IT Resource Management has not yet added support.

For %CPPLOT1, %CPPLOT2, and %CPSPEC, the corresponding procedure in SAS/GRAPH is PROC GPLOT.

For %CPCCHRT and CPCHART, the corresponding procedure in SAS/GRAPH is PROC GCHART.

For %CPG3D, the corresponding procedure in SAS/GRAPH is PROC G3D.

For %CPTABRPT, the corresponding procedure in the SAS System is PROC TABULATE.

For example, you might specify an annotate data set:

```
%CPPLOT1(....,
         PROCOPT=ANNO=WORK.MYANNO,
         ....);
```

The default value is blank (no value).

For more details about what options are valid on the PROC GPLOT, PROC G3D, and PROC GCHART statements, see your SAS/GRAPH documentation. For information about PROC TABULATE, see your SAS System documentation.

REDLVL=*PDB-level*

specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

SGRP_LAB=*label (obsolete; use LABELS=)*
specifies the label for the variable that is specified by the SUBGROUP= parameter. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify a label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

STAT=*statistic*
specifies what statistic to use to summarize data for the analysis variable(s) when the analysis variable (*variable-label-pairs*) is graphed and when the PERIOD= parameter is not ASIS. *The default statistic is MEAN*. Valid values are as follows:

MEAN
is the arithmetic average. Mean is one measure that is used to describe the center of a distribution of values. Other measures include mode and median.

SUM
is the sum of values for all observations.

STATLAB=*'label'*
specifies how to label the response axis if you do not want to use the name of the statistic. If you specify this parameter, then you should specify a label for the statistic that is specified in the STAT= parameter on this macro.

Note: You can specify only one label for the statistic. △

To use a label such as Monthly Total, you could specify the following:

```
STAT= sum,
STATLAB=Monthly Total
```

Do not include commas, equal signs, parentheses, or ampersands in the labels.

STMTOPT=*string*
where *string* is any text that is permitted as an option for the statement in the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro.

For %CPPLOT1 and %CPPLOT2, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPCCHRT and %CPCHART, the statement corresponds to the value of the TYPE parameter. For example, if TYPE=HBAR, then the statement will be the HBAR statement in PROC GCHART, and *string* will be added after the *'/'* in that statement.

For %CPSPEC, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPG3D, the corresponding statement in SAS/GRAPH is the SCATTER statement in PROC G3D.

For %CPTABRPT, the corresponding statement in SAS is the TABLE statement in PROC TABULATE.

If you want to customize the labeling and details of the axis for the group variable, you can specify an axis statement such as AXIS3 and then instruct %CPCHART to use it:

```
AXIS3 LABEL=(COLOR=BLUE);
%CPCHART(table,
         ....
         GROUP=gvar,
         STMTOPT=GAXIS=AXIS3,
         ....);
```

The default value is blank (no value).

For more details about what options are available for these statements, see the information about PROC GCHART, PROC G3D, and PROC GPLOT in your SAS/GRAPH documentation and PROC TABULATE in your SAS System documentation.

SUBGROUP=*variable*
specifies the name of a variable. This variable subdivides the groups that are specified by the value of the GROUP= parameter. Each subdivision becomes a section of a bar or block. The variable contains the same information as the STACKVAR= parameter, but the presentation is different. (In the %CPCCHRT macro, you can use the STACKVAR= parameter or the SUBGROUP= parameter, or neither, but not both.)

You cannot specify a SUBGROUP= variable if you specify TYPE= PIE or TYPE=STAR.

TABTYPE=INTERVAL | EVENT
specifies whether each observation in the data that is read into the table represents a specified interval of time or whether each observation was logged in response to

an event, such as when a job began or ended. *If the data is in a view or data set in the detail, day, week, month, or year level of the active PDB, then the default is the value that is specified in the table's definition in the active PDB's data dictionary.*

INTERVAL
each observation represents an interval of time

EVENT
each observation represents an event.

TOPN=*number*
specifies the maximum number of bars to chart. This produces a chart that displays the top *number* values (of the analysis variable as summarized by the STAT= parameter) and does not display lower values. For example:

- If ORDER=DESCENDING, and neither the GROUP= parameter nor the SUBGROUP= parameter is set, then TOPN=10 generates a chart of the 10 largest values of the analysis variable.
- If ORDER=DESCENDING, and the GROUP= parameter is set and the SUBGROUP= parameter is not set, then the 10 largest values of the analysis variable are found, without regard to group, and the 10 values are displayed within their groups.
- If ORDER=DESCENDING, and the GROUP= parameter and the SUBGROUP= parameter are both set, then the 10 largest values of the analysis variable are found, without regard to group and subgroup, and the 10 values are displayed within their groups and subgroups.

If you specify the TOPN= parameter, then the default value for the ORDER= parameter is DESCENDING; you can use ORDER=DESCENDING or ORDER=ASCENDING, but do not use ORDER=MIDPOINT. Specifying ORDER=ASCENDING and TOPN=*number* enables you to create a "BottomN" report of the lowest *number* values of the analysis variable.

TYPE=*chart-type*
is the type of chart to produce. *The default type is HBAR.* Valid values are as follows:

BLOCK
creates a three-dimensional block chart in which the height of each block represents the statistic for the analysis variable that is being charted.

HBAR
creates a horizontal bar chart in which the length of each bar represents the statistic for the analysis variable that is being charted.

HBAR3D
creates a horizontal bar chart like HBAR, but displays the 2-dimensional results in a way that simulates a 3-dimensional view.

PIE
creates a circular pie chart in which the size of each slice represents the statistic for the analysis variable that is being charted.

PIE3D
creates a circular pie chart like PIE, but displays the 2-dimensional results in a way that simulates a 3-dimensional view.

STAR
creates a star chart in which the size of each peak represents the statistic for the analysis variable that is being charted.

VBAR
creates a vertical bar chart in which the height of each bar represents the statistic for the analysis variable that is being charted.

VBAR3D
creates a vertical bar chart like VBAR, but displays the 2-dimensional results in a way that simulates a 3-dimensional view.

Note: If the value of the analysis variable is zero or missing for every slice (in a TYPE=PIE chart) or every peak (in a TYPE=STAR chart), then the report definition stops. You can use the WHERE= parameter to circumvent this, by specifying WHERE= *analysis-variable*>0. The values that are eliminated from pies and stars are ones that would not have been in the pie or star in any case. △

VREF=*value-list*
indicates values on the response axis where you want to draw one or more lines to use as reference lines. Separate the items in the *value-list* with one or more blanks.
value-list can also be in the form of:

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

The separator can be either a comma or blanks.

Note: This parameter is for horizontal bar charts, vertical bar charts, and block charts only. This parameter is ignored by pie charts and star charts. △

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.
Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.
This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.
For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WEIGHT=*weight-variable*

specifies the alternate variable by which to weight (multiply) statistical calculations. If no WEIGHT= variable is specified and the table type is INTERVAL, then the variable DURATION is used as the weight.

For example, if most observations have a value of 15 for the weight variable and one observation has a value of 30 for the weight variable, then that observation has twice the weight of each of the other observations in any calculations.

WHERE=*where-expression*

specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

YVAL=*number*

is the number of bars or blocks to print on one page of the graph. If TYPE= BLOCK, HBAR, HBAR3D, VBAR, or VBAR3D, then the YVAL= parameter represents the number of blocks or bars to print on one page. *The default number is based on the output device type*.

If the number of specified variables, groups, subgroups, or stack variables is greater than the value of YVAL=, then the chart is broken into multiple pages. For example, if you specify a group variable that has nine instances, such as nine different machines, and you set YVAL=6, then your graph prints on two pages.

The first page contains a graph with the first six bars or blocks, and the second page contains a graph with the last three bars or blocks.

This parameter is not meaningful when TYPE=PIE or STAR.

ZERO=YES | NO
specifies whether bars with a value of zero are displayed. This parameter is not required, and the default value is NO, which indicates that bars with a value of zero are not displayed. If you specify ZERO=YES, then bars with a value of zero are displayed.

The TYPE= parameter must be set to HBAR, HBAR3D, VBAR, or VBAR3D in order for this parameter to take effect.

%CPCHART Notes

If you are creating a PIE or STAR chart and the analysis variable for any of the groups (represented by pies or stars) have all values set to zero or missing, then the macro will stop. To avoid this problem, specify a where clause that contains *analvar > 0*.

There are several ways to create charts to analyze your data. The following list describes the different styles of charts that you can produce by using this macro.

- %CPCHART Style 1
  - Produces one pie/star or one set of bars/blocks.
  - In the pie or star, there is one slice or peak for each value of the class variable. In the set of bars or blocks, there is one bar or block for each value of the class variable.
  - The size of a slice or peak or the length of a bar or block represents the specified statistic of the analysis variable.
  - You must specify one class variable and one analysis variable.
- %CPCHART Style 2
  - Produces, for each value of the group variable, one pie/star or one set of bars/blocks.
  - In the pie or star there is one slice or peak for each value of the class variable. In the set of bars or blocks, there is one bar or block for each value of the class variable.
  - The size of a slice or peak or the length of a bar or block represents the specified statistic of the analysis variable.
  - You must specify one class variable, one analysis variable, and one group variable.
- For both styles you can specify one optional statistic and one optional weight variable. The statistic is used to collapse (summarize and replace) data if the Summary Time Period is not AS IS. The default statistic is MEAN for bar and block charts and SUM for star and pie charts. In a table of type interval, if no alternate weight variable is specified, then DURATION is used as the weight variable. (For each observation, the value of the weight variable is used to weight the values of the analysis variables during report-time summarization, such as the summarization when Summary Time Period is not AS IS.)
- For both styles you can also specify *n* optional variables by using the BY= parameter. No BY variables are required. If there is one BY variable, then a separate section of the report is generated for each value of the BY variable. If there is more than one BY variable, then a separate section of the report is generated for each unique combination of the values of the BY variables.

The variables that you specify in the *classname*, *variable-label-pairs*, BY=, FORMATS=, GROUP=, LABELS=, SUBGROUP=, and WEIGHT= parameters must exist in the table that is specified by the *dataset* parameter and in the level that is specified by the *dataset* or REDLVL= parameter. If the level is detail, then the variable must have a KEPT status of YES. If the level is day, week, month, or year, then the variable must be in the level's class variables list or ID variables list or must be a statistic that has been selected for calculation.

Refer to the %CPCCHRT macro for additional chart styles.

%CPENTCPY

Copies .SOURCE catalog entries to a directory or PDS

%CPENTCPY Overview

The %CPENTCPY macro is used to copy .SOURCE catalog entries in a SAS catalog to files in a UNIX or Windows directory or to members in a z/OS PDS.

%CPENTCPY Syntax

%CPENTCPY(
,**FROM=***entryname-list*
,**FROMCAT=***libref.catalog*
,**TO=***file-member-list*
,**TOLOC=***dir-PDS-name*
,**TYPE=***extension*
,<**FTPCNTL=**YES | NO>
<,**HTMLURL=***URL-specification*>
,<,**IMAGEURL=***URL-specification*>
<,**_RC=***macro-var-name*>
,<**REPLACE=**YES | NO>);

Details

FROM=*entryname-list*
lists the (one-level) entry names of the .SOURCE catalog entries that are to be copied from the catalog that is specified by the FROMCAT= parameter. If the list contains more than one name, then separate the names with one or more blanks. The total list length has a limit of 32,767 characters.

%CPENTCPY builds the four-level catalog entry names by using the value of the FROMCAT= parameter as the first two levels and the value SOURCE as the fourth level.

This parameter is required.

Note: This list is sensitive to position. Each name in the FROM= parameter's list corresponds to the name in the same position in the TO= parameter's list. △

FROMCAT=*libref.catalog*
specifies the two-level name of the SAS catalog whose catalog entries are to be copied. The libref must already be defined by the time that you call this macro. One way to define the libref is to use the SAS LIBNAME statement. For more

information about the SAS LIBNAME statement on your platform, see the SAS Companion for your current release of SAS.

This parameter is required.

TO=*file-member-list*
specifies the list of names to use for the source entries in the TOLOC= output location. *This parameter is required.*

If the value of TOLOC= is a UNIX or Windows directory, then the value of this parameter is a list of filenames (without period or extension) to use in that directory. (For the extension to be used, see the TYPE= parameter.) If the value of TOLOC= is a PDS, then the value of this parameter is a list of member names to use in that PDS.

If the files or members do not exist in the TOLOC= location, then they will be created. If the files already exist, then by default they are replaced. If you do not want to replace the files, then specify REPLACE=NO.

If the list contains more than one name, then separate the names with one or more blanks. *This is a required parameter.*

Note: This list is sensitive to position. Each name in the TO= list corresponds to the name in the same position in the FROM= list. △

TOLOC=*dir-PDS-name*
specifies the full path and name of the directory or the full name of the PDS in which the files or members (specified in the TO= parameter) are to be created. This directory or PDS must already exist.

This parameter is required.

If you are copying members to a PDS, then see also the FTPCNTL= parameter.

TYPE=*extension*
specifies the extension that is to be appended to the filenames if you are copying to a directory-based file system (such as UNIX, Windows, or the z/OS UNIX File System). Do not include the period that separates the filename and the extension. The file extension is used in directory systems to determine the format of the data in the file. For example, files with an extension of .htm contain text with html tags.

This parameter is required if you are copying to a directory-based file system. Otherwise, this parameter is ignored (for example, if you are copying to a z/OS file system).

FTPCNTL=YES | NO
specifies whether to create the necessary control information that will be needed if you later transmit the copied entries to a remote host by using the %CPFTPSND macro. This parameter is used only if you are copying to a PDS (on z/OS). *The default value is NO.*

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **`welcome.htm`**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**`welcome.htm`** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **`HTMLURL=http://www.reporter.com/reports`** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **welcome.htm** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

_RC=*macro-var-name*

specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

REPLACE=YES | NO

specifies whether to overwrite an existing file or member if that file or member already exists. If you specify REPLACE=NO, then the file or member that you are copying will not overwrite the existing file or member. If you specify REPLACE=YES, then the existing file is deleted and a new file is create with the same name.

The default is REPLACE=YES.

%CPENTCPY Notes

In the directory or PDS to which you are copying, you must have the appropriate access permissions to create (if necessary) and write to the files or members.

%CPENTCPY Example

This example copies YOURHELP.SOURCE and MYNOTES.SOURCE from the catalog WORK.MYCAT to the directory /u/myid/mywebout as help.htm and notes.htm.

```
%cpentcpy( from=yourhelp mynotes,
           fromcat= work.mycat,
           to=help notes,
           toloc=/u/myid/mywebout,
           type=htm );
```

%CPEXCEPT

Evaluates exception rules in a specified folder

%CPEXCEPT Overview

The %CPEXCEPT macro is used to evaluate all the exception rules in the specified folder (rule data set).

The interactive exception reporting system enables you to identify exception conditions in your data through the creation and evaluation of exception rules. Exception rules must be defined through the graphical user interface, but they can be evaluated on a periodic (one time a day, one time a week, and so on) basis in batch, by using this macro.

Evaluating an exception rule involves examining the specified data to see whether there are exceptions as defined in the rules.

Exception rule definitions are stored in an exception rule folder, RULEDS=*rule-folder-dataset*. Exception rule results are stored in the exception results folder (RESULTS=*exception-result-folder*).

%CPEXCEPT Syntax

%CPEXCEPT(
RULEDS=*libref.name*
<,**PRINT=**LOG | PRINT>
<,**_RC=***macro-var-name*>
<,**RESULTS=***exception-result-folder*>
<,**STOPACT=**STOP | ABORT>
<,**STOPCNT=***n*>);

Details

RULEDS=*libref.name*
specifies the location of the folder that contains the exception rule definitions, which will be used (by %CPEXCEPT or by %CPXHTML) to find the exceptions that are recorded in the RESULTS= file.

Note: The %CPEXCEPT macro runs *before* the %CPXHTML macro runs. △

The folder is implemented as a SAS data set. *This parameter is required for the %CPEXCEPT macro. It is also required for the %CPXHTML macro unless the*

GENERATE= parameter is set only to FOLLOWUP. (The default for the GENERATE= parameter is ALL, but you can specify a single type or any combination of types.)

Identify the data set by using the two-level format *libref.name*, where *libref* is already defined and *name* is the name of your exception rule data set.

If the libref is not already defined in your SAS session, then you must define the libref (by using a SAS LIBNAME statement) before you run this macro. You must have at least read access to the library that is referenced by the libref.

Note: %CPXHTML determines which exception rules are to be reported as "service level" rules by checking the third through fifth characters of the rulename for "SLA". If these characters are "SLA", then the rule is treated as a service-level rule and it appears in the service-level report that is produced by the GENERATE=SERVICE (or GENERATE=ALL) setting in %CPXHTML.

In addition, the RULEDS= parameter assigns a title to the exception report that is produced by the GENERATE=SUMMARY (or GENERATE=ALL) setting in %CPXHTML. The title of the exception report is the description that was associated with the RULEDS= data set when the data set was created. △

PRINT=LOG | PRINT
specifies whether and where to print the results from evaluating the rule definitions. The results are the contents of the folder that is specified by the RESULTS= parameter.

LOG
prints the output to the LOG window

PRINT
prints the output to the OUTPUT window.

If the PRINT= parameter is not specified, results are not printed.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

RESULTS=*exception-result-folder*
specifies the SAS data set name of the folder that is to contain the exceptions that are found by the %CPEXCEPT macro. Identify the data set by using the two-level name *libref.datasetname*, where libref is already defined. You can define the libref by using a SAS libname statement before you run this macro.

The exception Web pages that are produced by %CPXHTML are generated from this folder of exceptions. The initial default folder is SASUSER.XRESULTS. After the initial use, the default folder is the most recent folder that is used for results either by the %CPEXCEPT or %CPXHTML macro or by the exception subsystem in the GUI.

For the %CPEXCEPT macro,

- the results data set can reside in any existing SAS library to which you have write access, and it can have any name that is unique to the library.
- the data set is created automatically if it does not already exist. (The data set is cleared automatically if it already exists and if it contains any exceptions from a previous run.)

STOPACT=STOP | ABORT
specifies what action to take when the exception count limit that is specified by STOPCNT= is reached.

STOP
stops evaluating the exception rules.

ABORT
stops evaluating the exception rules and issues a SAS ABORT request. *Specifying the ABORT option can have very serious consequences*. See the *SAS Language Reference* for a complete discussion of the ABORT statement, the results of which vary among operating environments.

The default is STOPACT=STOP.

STOPCNT=*n*
specifies a limit to the number of exceptions. If more exceptions are found, then evaluation of the rules is discontinued.

See the STOPACT= parameter for information about specifying what specific action is taken when the specified limit is reached.

The default is 999.

%CPEXCEPT Notes

The results of %CPEXCEPT's exception search can be viewed on the Web by running the %CPXHTML macro in batch mode. The %CPXHTML macro uses the *exception-result-folder* that is produced by the %CPEXCEPT macro as one of its input data sets. (The %CPEXCEPT macro runs *before* the %CPXHTML macro.) For more information, see the macro documentation for "%CPXHTML" on page 536.

For a rule to be evaluated, the following conditions must be met:

- If the data is specified as *level-name.PDB-table-name*, where *level-name* is detail, day, week, month, or year, then the table must be in the active PDB.
- Otherwise, the data is specified as *libref.data-set-name* or *data-view-name*. The libref must be defined before %CPEXCEPT is called.

%CPEXCEPT Example

This example evaluates the rules that are defined in SASUSER.XRULES against the specified data and writes information (about any exceptions that are found) to SASUSER.XRESULTS.

```
%CPEXCEPT( ruleds=SASUSER.XRULES,
           results=SASUSER.XRESULTS );
```

%CPG3D

Produces a three-dimensional plot

%CPG3D Overview

The %CPG3D macro produces a three-dimensional (3-D) plot that displays multidimensional trends or concentrations in data.

You can produce several styles of 3-D plots:

- One style has an x,z plot at each tick mark along the Y axis, with one x,z plot for each value of a class variable. Each x,z plot has a different color.

  You can specify the X variable, Z variable (analysis variable), and class variable.
- One style has an x,z plot at each tick mark along the Y axis, with one x,z plot for each analysis variable. Each x,z plot has a different color.

  You can specify the X variable and two to fifteen Z variables (analysis variables).
- One style is an x,y,z plot of values of the X, Y, and Z variables.

  You can specify the X variable, Y variable, and Z variable.

For more information about the styles and related information, see the Notes section.

For more information about the underlying procedure, refer to the G3D procedure (PROC G3D) in the SAS/GRAPH documentation for your current release of SAS.

%CPG3D Syntax

%CPG3D(
dataset
,*variable-label-pairs*
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***BY-variable-list*>
<,**CL_LAB=***label (obsolete; use LABELS=)*>
<,**CL_NAM=***variable-name*>
<,**CTEXT=***color*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**NEEDLE=**NO | YES>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**PERIOD=***time-interval*>
<,**PROCOPT=***string*>
<,**REDLVL=***PDB-level*>

<,**ROTATE=***angle-list*>
<,**SMALLDEV=***device-driver*>
<,**STAT=***statistic*>
<,**STMTOPT=***string*>
<,**TABTYPE=**INTERVAL | EVENT>
<,**TILT=***angle-list*>
<,**TYPE=***plot-type*>
<,**WEBSTYLE=***style*>
<,**WEIGHT=***weight-variable*>
<,**WHERE=***where-expression*>
<,**XLAB=***label*> (obsolete; see LABELS=)
<,**XVAR=***variable*>
<,**YLAB=***label*> (obsolete; see LABELS=)
<,**YVAR=***variable*>);

Details

dataset
specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

variable-label-pairs
specifies a list of pairs. (For %CPCHART, you can use only one pair.) Each pair consists of the name of one analysis variable and the label for that analysis variable. Each label has a maximum length of 16 characters.

The maximum number of pairs that you can specify is 15, depending on the type of report that you are running. *For a detailed explanation of the variables that are displayed on each axis and the maximum number of variables that can be specified for the report, refer to the "Overview" section of the macro description.*

If you specify multiple analysis variables, then the unit of measure for all of the analysis variables must be the same and the range of values should be similar.

The analysis variable is typically displayed on the left (Y) axis. However, certain combinations of parameters might affect the axis on which the analysis variable is displayed or the number of variables that can be specified, depending on the report macro that you use.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of a label contains an unmatched

single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in a label.

You are required to specify at least one variable in this positional parameter. However, you are not required to specify labels when you use this parameter. In fact, using the LABELS= parameter to specify the labels is preferred. Because this is a positional parameter, you must use a comma as a placeholder for the label when you specify more than one variable but do not specify labels. For example, to specify three variables and no labels, you could specify **`var1,,var2,,var3,KEYWORD=...`**. In this example, you have specified variable 1 but no label for variable 1, variable 2 but no label for variable 2, and variable 3 but no label for variable 3. You then specify any keyword parameters, such as the LABELS= parameter, that you want to use in the report definition.

You can specify the labels by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the LABELS= parameter, then any labels that are specified by using this parameter are ignored. If labels are not specified by using this parameter and not specified by using the LABELS= parameter, then the variables' labels in the data dictionary are used.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **`dd:mmm:yyyy:hh:mm:ss`**, where **`dd`** is day, **`mmm`** is month, **`yyyy`** is year (in two-digit or four-digit notation), **`hh`** is hour (using a 24-hour clock), **`mm`** is minute, and **`ss`** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **`dd:mmm:yyyy`** AT **`hh:mm:ss`**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **`hh:mm:ss`**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **`hh:mm:ss`** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.

- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*BY-variable-list*

lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB

□ the class variables list in the specified level of the specified table in the PDB.

CL_LAB=*label (obsolete; use LABELS=)*
specifies the label for the class variable. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

CL_NAM=*variable-name*
specifies the name of the class variable for the report. For each unique value of the class variable, the macro creates a separate portion or group on the graph.

Compare this type of variable to the variable specified by the macro's BY=parameter. The BY variable creates separate graphs for each unique value of the variable. For example, the following combination of parameters produces one report or plot for each unique value of the BY variable MACHINE:

```
cl_nam=diskid,
by=machine
```

Each of the graphs for MACHINE includes all possible values of DISKID, the class variable. If you are producing a plot, then each unique value of DISKID is represented by a different color or plot symbol. If you are producing a chart, then each value of DISKID is represented by a different column or row.

CTEXT=*color*
indicates the color of the text on a plot. *The default text color is green*. You can specify the background color by using the CBACK= parameter in the SAS GOPTIONS statement. The default background color is device dependent. Refer to the information on your output device in "Using Graphics Devices" in the SAS/GRAPH documentation for your current release of SAS.

If the CTEXT= parameter is blank, then a color specification for the axis color is searched for in this order:

1. the CTEXT= option in a GOPTIONS statement
2. the default, which is the first color in the color list specified in the GOPTIONS COLORS= statement or the default list of colors for the device.

Note: If the color of the text is the same as the color of the background, then the text appears to be missing. △

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*

lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see “Global and Local Macro Variables” on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are

directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file

expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **`&`** obtains the value of the

macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)

specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*

specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their

Web browsers, they see a larger version of the graph report. The larger version is static.

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).

- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*
specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.
- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

NEEDLE=NO | YES

specifies whether plot points have lines that connect the points to the X-Y plane. If you specify NEEDLE=NO, then you produce a three-dimensional scatter plot. NEEDLE=YES connects the plot points to the plane by using lines. *The default is NEEDLE=YES when the X variable is DATETIME; otherwise, the default is NEEDLE=NO*.

See the TYPE= parameter for examples of the symbols that can be used for scatter plots.

Note: NEEDLE=NO has no effect on prism or pillar plots. △

OUTDESC=*output-description*

if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*

- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*

for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT*: WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT

macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename | entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT*: G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

PALETTE=*palette-name*

specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **`sasuser.palette.win`** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PERIOD=*time-interval*
is the summarization period for the report. *The default is ASIS*. For values other than ASIS, the values for the time period within a class or category are combined (as specified in the STAT= parameter) to form a single point on a plot, a bar on a chart, or a row or column in a table.

The following list provides the period name, the effect that it has on the report, and an example of how to specify the period.

- ASIS - leaves datetime in its present form. Example: 09Jun2003:14:37:25
- 15MIN - transforms the time to the first minute of the 15-minute period in the hour and retains the date. Example: 09Jun2003:14:30
- HOUR - transforms the time to the first minute of the hour and retains the date. Example: 09Jun2003:14:00
- 24HOUR - transforms the time to its hour component and omits the date. Range: 0–23. Example: 14
- DATE - omits the time and retains the date. Example: 09Jun2003
- WHOLEDAY - obsolete; use DATE.
- WEEKDAY - transforms the day to the day of the week (where 1=Sunday, 2=Monday, and so on) and omits the month, year, and time. Range: 1–7, which displays as Sunday through Saturday. Example: Monday
- MONTHDAY - omits everything but the day of the month. Range: 1–31. Example: 9

- YEARDAY - transforms the day to the day of the year and omits the month, year, and time. Range: 1–366. Example: 160
- WEEK - transforms the date to the first day of the week as defined by Start-of-Week and omits the time. (Start-of-Week is a PDB property that you can specify with the %CPPDBOPT macro. By default, Start-of-Week is Sunday.) Example: 08Jun2003
- MONTH - omits the day and the time. Example: Jun2003
- YEARMONTH - omits everything but the month of the year. Range: 1–12. Example: 6
- QTR - transforms the month to the first month of the quarter and omits the day and time. Example: Apr2003

Note: You can change the width of the bar that appears on the 3-dimensional plot generated by %CPSPEC. If you specify PERIOD= ASIS, or 15MIN, or HOUR, the bar will be thinner in order to accommodate the increased number of points per line that is typical with shorter periods. △

PROCOPT=*string*
where *string* is any text that is permitted on the PROC statement of the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro. This allows expert SAS/GRAPH users to take advantage of options that are not supported by SAS IT Resource Management macros or new options for which SAS IT Resource Management has not yet added support.

For %CPPLOT1, %CPPLOT2, and %CPSPEC, the corresponding procedure in SAS/GRAPH is PROC GPLOT.

For %CPCCHRT and CPCHART, the corresponding procedure in SAS/GRAPH is PROC GCHART.

For %CPG3D, the corresponding procedure in SAS/GRAPH is PROC G3D.

For %CPTABRPT, the corresponding procedure in the SAS System is PROC TABULATE.

For example, you might specify an annotate data set:

```
%CPPLOT1(....,
         PROCOPT=ANNO=WORK.MYANNO,
         ....);
```

The default value is blank (no value).

For more details about what options are valid on the PROC GPLOT, PROC G3D, and PROC GCHART statements, see your SAS/GRAPH documentation. For information about PROC TABULATE, see your SAS System documentation.

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △
- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL=

parameter that was specified in the underlying report definition(s) being invoked.

ROTATE=*angle-list*
specifies one or more angles at which to rotate the X-Y plane about the perpendicular Z axis. A separate graph is produced for each angle that is listed. The units of measure for *angle-list* are degrees. *The default value is -20*.

The *angle-list* values can be specified in the following ways:

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
   <BY increment> <n . . . n>
```

The values that are specified in *angle-list* can be negative or positive and can be greater than 360 degrees. For example, a rotation angle of 45 degrees can also be expressed as

```
rotate=45
rotate=405
rotate=-315
```

Specifying a sequence of angles produces a graph for each pair of rotate-and-tilt angles. The angles that are specified in the ROTATE= option are paired with any angles that are specified with the TILT= parameter. Therefore, you must specify the same number of angle values for both the ROTATE= and TILT= parameters.

For more information about the *angle-list* value, see the ORDER= parameter in the "AXIS Statement Options" section of the SAS/GRAPH documentation for your current release of SAS.

For more information about rotating the plot, see the SAS/GRAPH documentation for your current release of SAS. Refer to *Example 2: Rotating a Surface Plot* in *The G3D Procedure*.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

STAT=*statistic*
specifies what statistic to use in order to summarize data that spans the time interval that is specified by PERIOD=. The selected statistic is used for all variables in the report. *The default statistic is MEAN*. The STAT= parameter is not valid if PERIOD=ASIS. Valid values for STAT= are as follows:

CSS
is the sum of squares, corrected for the mean.

CV
is the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAX
is the maximum value for all observations.

MEAN
is the arithmetic average. Mean is one measure that is used to describe the center of a distribution of values.

MIN
is the minimum value for all observations.

N
(or COUNT) is the number of observations with nonmissing values for the variables that are being summarized.

NMISS
the number of observations with missing values for the variable being summarized.

RANGE
is the maximum of the difference between the maximum and minimum values for the variables, RANGE=MAX-MIN.

STD
is the standard deviation or square root of variance. Like the variance, it is a measure of the dispersion about the mean, but in the same unit of measure as the data.

STDERR
is the positive square root of the variance of a statistic.

SUM
is the sum of values for all observations.

USS
is the uncorrected sum of squares.

VAR
is a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: If you specify the BY= parameter, the statistics are calculated separately for each value of the BY variable or for each unique combination of the BY variable values, if you specify multiple BY variables. △

STMTOPT=*string*
where *string* is any text that is permitted as an option for the statement in the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro.

For %CPPLOT1 and %CPPLOT2, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPCCHRT and %CPCHART, the statement corresponds to the value of the TYPE parameter. For example, if TYPE=HBAR, then the statement will be the HBAR statement in PROC GCHART, and *string* will be added after the '/' in that statement.

For %CPSPEC, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPG3D, the corresponding statement in SAS/GRAPH is the SCATTER statement in PROC G3D.

For %CPTABRPT, the corresponding statement in SAS is the TABLE statement in PROC TABULATE.

If you want to customize the labeling and details of the axis for the group variable, you can specify an axis statement such as AXIS3 and then instruct %CPCHART to use it:

```
AXIS3 LABEL=(COLOR=BLUE);
%CPCHART(table,
         ....
         GROUP=gvar,
         STMTOPT=GAXIS=AXIS3,
         ....);
```

The default value is blank (no value).

For more details about what options are available for these statements, see the information about PROC GCHART, PROC G3D, and PROC GPLOT in your SAS/GRAPH documentation and PROC TABULATE in your SAS System documentation.

TABTYPE=INTERVAL | EVENT
specifies whether each observation in the data that is read into the table represents a specified interval of time or whether each observation was logged in response to an event, such as when a job began or ended. *If the data is in a view or data set in the detail, day, week, month, or year level of the active PDB, then the default is the value that is specified in the table's definition in the active PDB's data dictionary.*

INTERVAL
each observation represents an interval of time

EVENT
each observation represents an event.

TILT=*angle-list*
specifies one or more angles at which to tilt the graph toward you. A separate graph is produced for each angle that is listed. *The default tilt is 80*. The units for *angle-list* are degrees. Specify *angle-list* by using the same format that is described for ROTATE=. The values for *angle-list* must be in the range of 0 through 90. Refer to the description of the ROTATE= parameter on this macro for an explanation of the interaction between ROTATE= and TILT=.

For more information about tilting the plot, see the SAS/GRAPH documentation for your current release of SAS. Refer to *Example 3: Tilting Surface Plot"* in *The G3D Procedure*.

TYPE=*plot-type*
specifies one of the following shapes in order to represent the analysis variable on the graph.

Table 3.1 Shape Values

BALLOON	CLUB	CROSS
CUBE	CYLINDER	DIAMOND
FLAG	HEART	PILLAR

POINT	PRISM	PYRAMID
SPADE	SQUARE	STAR

The default value is BALLOON.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WEIGHT=*weight-variable*
specifies the alternate variable by which to weight (multiply) statistical calculations. If no WEIGHT= variable is specified and the table type is INTERVAL, then the variable DURATION is used as the weight.

For example, if most observations have a value of 15 for the weight variable and one observation has a value of 30 for the weight variable, then that observation has twice the weight of each of the other observations in any calculations.

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE.

(*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

XLAB=*label (obsolete; see LABELS=)*
specifies the label for the X axis of a plot. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the XVAR= variable's label in the PDB's data dictionary is used.

XVAR=*variable*
specifies the name of the variable for the X axis of a plot. The variable must be numeric. If you do not provide a variable's name, then the variable's name defaults to DATETIME.

YLAB=*label (obsolete; see LABELS=)*
specifies the label for the Y axis of a plot. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the YVAR= variable's label in the PDB's data dictionary is used.

YVAR=*variable*

specifies the name of the variable for the third axis of a %CPG3D plot. The variable must be numeric.

%CPG3D Notes

You can use this macro to produce three styles of reports:

- %CPG3D Style 1
  - There is one X variable. The name of the variable for the horizontal axis (X axis) is specified by the XVAR= parameter.
  - There is one analysis variable. The analysis variable corresponds to the vertical axis (Z axis). The name of the analysis variable is specified by the first (and only) variable in the *variable-label-pairs* parameter.
  - There is one class variable. The class variable corresponds to the "diagonal" axis (Y axis). The name of the class variable is specified by the CL_NAM= parameter.
  - Note: Do not specify the YVAR= parameter. Specify only one variable in the *variable-label-pairs* parameter.
  - The result is one x,z plot for each value of the CL_NAM= parameter, with the x,z plots occurring at regular tick marks along the Y axis. Each plot is labeled with its value of the CL_NAM= parameter. Each plot uses a different color. The height of a data point represents the value of the analysis variable being plotted on the x,z plot.

    For example, if each value of the CL_NAM= parameter is a machine name, there will be one x,z plot for each machine name, with the plots at regular intervals along the "diagonal" axis.
- %CPG3D Style 2
  - There is one X variable. The name of the variable for the horizontal axis (X axis) is specified by the XVAR= parameter.
  - There are two to fifteen analysis variables. The analysis variables correspond to the vertical axis (Z axis). The names of the variables for the vertical axis (Z axis) are specified by the variables in the *variable-label-pairs* parameter. All of the variables in the *variable-label-pairs* parameter should have the same unit of measurement.
  - There are no class variables.
  - Note: Do not specify the YVAR= parameter. Do not specify the CL_NAM= parameter (class variable).
  - The result is one x,z plot for each variable in the *variable-label-pairs* parameter, with the x,z plots occurring at regular tick marks along the Y axis. Each plot is labeled with the name of its Z variable. Each plot uses a different color. The height of a data point represents the value of the analysis variable being plotted on that x,z plot.

    For example, suppose that XVAR=DATETIME and that the *variable-label-pairs* parameter is **`INPC,,OUTPC,,ABNPC,`**, where INPC is the count of incoming calls, OUTPC is the count of outgoing calls, and ABNPC is the count of abandoned calls for a given phone line (trunk). The result is three plots, each one offset from the next along the Y axis: one plot of INPC (vertical) against DATETIME (horizontal), one plot of OUTPC (vertical) against DATETIME (horizontal), and one plot of ABNPC (vertical) against DATETIME (horizontal).
- %CPG3D Style 3

- There is one X variable. The name of the variable for the horizontal axis (X axis) is specified by the XVAR= parameter.
- There is one analysis variable for the Z axis. The name of the variable for the vertical axis (Z axis) is specified by the first variable in the *variable-label-pairs* parameter.
- There is one analysis variable for the Y axis. The name of the variable for the "diagonal" axis (Y axis) is specified by the YVAR= parameter.
- There is no class variable.
- Note: Do not specify more than one variable in the *variable-label-pairs* parameter. Do not specify the CL_NAM= parameter.
- The result is one x,y,z plot.
- The GUI does not support setting a Y-variable. Therefore, to use this style, you must invoke the %CPG3D macro directly.

Related information:

- For each of the report styles, you can specify *n* optional BY variables. No BY variables are required. If there is one BY variable, then a separate 3-D graph is generated for each value of the BY variable. If there is more than one BY variable, then a separate 3-D graph is generated for each unique combination of values of the BY variables.
- For each of the report styles, you can specify one optional statistic. The statistic is used to collapse (summarize and replace) data if the Summary Time Period is not AS IS. The default statistic is MEAN.
- For each of the report styles, you can specify one optional weight variable. In a table of type interval, if no weight variable is specified, then DURATION is used as the weight variable.

%CPHTREE

Provides organization and navigation for your Web-enabled reports

%CPHTREE Overview

The %CPHTREE macro creates a tree structure and hierarchical view for the Web reports. (The Web reports are images and HTML output. The files are generated in batch mode when you specify OUTMODE=WEB in the report macros and are generated by the GUI when you select **`Web`** in the Report Output Options window.)

With one call to %CPHTREE, you can create directories that are displayed in your Web browser as report galleries, and their navigational files, which are displayed in your Web browser as an explorer-type view of the galleries. This set (the directories and navigational files) is referred to as a single tree, and the associated explorer view is referred to as a single-tree explorer view. Typically, the single tree is used for the reports on data from one data source.

With multiple calls to %CHPTREE, you can create a single tree for each of your data sources. If you create all the single trees under the same "super" directory, then the result is both a tree of single trees and an explorer-type view that is sometimes called a super-tree explorer view. The super-tree explorer view enables you to navigate from one single tree to another.

If you use the QuickStart Wizard to create batch jobs that will process, reduce, and create reports on your data, then a tree that is created by %CPHTREE (when the

Wizard runs the xRPTSTR.sas job) is used when you run the reporting job (xREPORT.sas).

%CPHTREE Syntax

%CPHTREE=(
,**CAT=***source-entry-name*
,**SUPERLOC=***directory-name*
<,**BIGSPACE=***amount-of-space*>
<,**DSPARMS=***dsparms-parameters*>
<,**LISTCLR=**YES | NO | QUICKSTART | REMAP>
<,**MAXLINES=***number-of-lines*>
<,**PTITLE1=***ptitle1-text*>
<,**PTITLE2=***ptitle2-text*>
<,**PTITLE3=***ptitle3-text*>
<,**_RC=***macro-var-name*>
<,**SEPARATR=***separator-character*>
<,**SMLSPACE=***amount-of-space*>
<,**SUBLOC=***directory-name*>
<,**TITLE1=***title1-text*>
<,**TITLE2=***title2-text*>
<,**TITLE3=***title3-text*>
<,**WEBSTYLE=***style*>);

Details

CAT=*source-entry-name*
specifies the four-level name of a source entry in a SAS catalog. For example, CAT=WORK.TEMP.TREE.SOURCE refers to the TEMP catalog in the WORK library and the TREE.SOURCE entry in that catalog. Additional information about the catalog entry is provided in the following three sections:

- *The Contents of the Catalog Entry*. The catalog entry contains an "outline." The "outline" describes the tree that you want to create, in terms of topics and groupings of topics and the directories that correspond to them. The following is an example of an outline:

```
0 Overview              ! pservov
0 Utilization
  1 Top Systems
    2 Processor         ! pservtc
    2 Memory            ! pservtm
    2 Storage           ! pservts
    2 Network           ! pservtn
  1 Critical Systems
    2 Processor         ! pservcc
    2 Memory            ! pservcm
    2 Storage           ! pservcs
    2 Network           ! pservcn
    2 Bottlenecks       ! pservcb
0 Service
  1 Availability      ! pservsa
  1 Responsiveness    ! pservsr
  1 Throughput        ! pservst
```

```
  1 Exception Summary ! pservex ! srvwelco.htm ! srvinfo.htm
0 Forecasts
  1 Resource Forecasts ! pserv4r
  1 Service Forecasts  ! pserv4s
0 Exceptions
  1 Exception Summary  ! pservex ! allwelco.htm ! allinfo.htm
  1 Exception Results  ! pservex ! excwelco.htm ! excinfo.htm
  1 Exception Reports  ! pservex
```

When you run the %CPHTREE macro, a tree is created by using the outline in the entry.

- *The Layout of a Line in the Catalog Entry.* The maximum length of a line is 200 characters. The line must not wrap. The line must not have any tab characters; to approximate tabs, you can use blanks. The first line must start with level 0. There are two kinds of line.

  The first kind of line represents a click on the selection path but not the end of a selection path. For example, the "0 Utilization" line and the "1 Top Systems" line are of this kind. The lines represent logical levels, but because they are groupings of items below them, they do not themselves represent a directory. The format for this kind of line is as follows:

  1. The first field is a number that indicates the logical level. Level 0 is the level below the directory that is specified by the SUBLOC= parameter; level 1 is the level below that, and so on. An indefinite number of levels are available, but 10 is a sensible limit. One or more blanks separate the first and second fields.
  2. The second field is a text string that describes the grouping of topics that the line represents.

  The second kind of line represents the end of a selection path. For example, the "0 Overview" line and the "2 Processor" lines are of this kind. The lines represent logical levels and, because they are not a grouping of items below them, also represent directories. The format for this kind of line is as follows:

  1. The first field is a number that indicates the logical level. Level 0 is the level below the directory that is specified with the SUBLOC= parameter; level 1 is the level below that, and so on. An indefinite number of levels are available, but 10 is a sensible limit. One or more blanks separate the first and second fields.
  2. The second field is a text string that describes the topic that the line represents. The topic labels the directory in the single-tree explorer type of view.
  3. The separator character (see the SEPARATR= parameter) separates the second and third fields.
  4. The third field is the name of the directory that is to contain the reports that are indicated by the description in the second field. For UNIX or Windows environments or for z/OS when using the z/OS UNIX File System area, this should be only the directory name and not a complete directory path. The directory will be created immediately under the directory that is defined by the SUPERLOC= parameter.

     On z/OS if you are not using the z/OS UNIX File System area, the third field is the right-most qualifier of the PDS that is to be created to contain the reports that are indicated by the description. The value of the SUPERLOC= parameter is used for the high-level qualifiers in the name.

This is the directory (without its pathname) or PDS (without its high-level qualifiers) that you specify as HTMLDIR= when you direct reports to this location. (%CPHTREE assumes that you will not specify IMAGEDIR= when you direct reports to this location.)

5 The separator character (see the SEPARATR= parameter) separates the third and fourth fields.

6 The fourth field is the name of the directory's file that you want the browser to display in the right (white) frame, when the topic is selected in the single-tree explorer view. If the directory will contain reports that are based on rules (that is, reports that are generated by %CPXHTML), then you must use one of these four files: allwelco.htm, excwelco.htm, srvwelco.htm, or welcome.htm. If the directory contains reports that are not based on rules (that is, reports that are generated by %CPRUNRPT, %CPPLOT1, %CPPLOT2, and so on), then typically the only such file is welcome.htm. If you leave this field empty, then welcome.htm is the default value. (On z/OS if you are not using the z/OS UNIX File System area, the fourth field is the name of the member in the PDS that has the same role.)

7 The separator character (see the SEPARATR= parameter) separates the fourth and fifth fields.

8 The fifth field is the name of the directory's file that the file in the fourth field displays in the left/blue frame when the topic is selected in the single-tree explorer view. Typically, if the fourth field is allwelco.htm, then use allinfo.htm; for excwelco.htm, use excinfo.htm; for srvwelco.htm, use srvinfo.htm; and for welcome.htm, use info.htm. If you leave this field empty, then info.htm is the default value. (On z/OS if you are not using the z/OS UNIX File System area, the fifth field is the name of the member in the PDS that has the same role.)

For the number of lines in the "outline," see the MAXLINES= parameter.

□ *Creating and Editing the Catalog Entry.*

Interactively, you can create the outline by typing it into the SAS Program Editor window and then saving it to the catalog entry by issuing a command in this form:

```
save libref.catalog_name.entry_name.source
```

The library to which the libref refers must already exist. (For example, you could use the PDB's ADMIN library and ADMIN libref. The %CPSTART macro automatically defines the librefs for the libraries in the PDB that is specified by the PDB= parameter.) The other three parts of the name are created if they do not already exist.

In batch or background mode, you can create the outline by using two calls to the %CPCAT macro. The first call loads the "outline" into a buffer. The second call loads the contents of the buffer into a catalog entry. The library to which the libref refers must already exist. (For example, you could use the PDB's ADMIN library and ADMIN libref. The %CPSTART macro automatically defines the librefs for the libraries in the PDB that is specified by the PDB= parameter.) The other three parts of the name are created if they do not already exist. For an example, see Example 6 for this macro.

SUPERLOC=*directory-name*

specifies the name of the "super" directory. When the "super" directory is created, a file that is named **`welcome.htm`** and a directory that is named **`WEB`** will be created in the super directory. Also, each directory that is specified in the outline that is in

the entry that is specified by the CAT= parameter uses the value of SUPERLOC= as its pathname. (Note that the use of a common pathname means that the directories are all immediately under the "super" directory. It is the explorer type of view that makes their contents appear to be organized as a hierarchical tree.)

Note: The former name of this parameter on this macro was OUTLOC=. That name is now obsolete for the %CPHTREE macro. △

For UNIX or Windows environments or for z/OS using the z/OS UNIX File System area, this parameter should specify the complete path or directory name for the "super" directory.

On z/OS when the z/OS UNIX File System area is not being used, this parameter specifies the name of the "super" PDS. After the "super" PDS is created, a member that is named WELCOME is automatically added in the super PDS. Also, an additional PDS is created whose name is built by appending WEB, as the right-most qualifier, to the name of the "super" PDS. For example, if the super PDS is named SUPER.PDS, then the welcome member would be named SUPER.PDS(WELCOME) and the additional PDS would be named SUPER.PDS.WEB.

BIGSPACE=*amount-of-space*

specifies the units of primary space and, optionally, secondary space and directory blocks that are required for each PDS that contains reports. *This parameter is optional if running on z/OS when the z/OS UNIX File System area is not being used; for other operating environments or for z/OS when the z/OS UNIX File System area is being used, this parameter is ignored.*

On z/OS, when the z/OS UNIX File System area is not being used, the trees are first made with PDSs (by a call to %CPHTREE) and then "copied" to the corresponding directories on UNIX or Windows (by a call to %CMFTPSND). This parameter applies to the PDSs that are in the "outline." The PDSs that contain reports typically require multiple cylinders of disk space. For example,

```
BIGSPACE=%QUOTE((CYL,(1,1,10)))
```

will result in PDSs that have allocations with the following:

```
SPACE=(CYL,(1,1,10))
```

The default is

```
BIGSPACE=%QUOTE((CYL,(2,5,96)))
```

The remaining parameters of the allocation are specified in the DSPARMS= parameter.

Note: The SAS %QUOTE function protects the contents as a string. That is, it protects the contents from any attempts by the SAS macro compiler to parse the string and then try to evaluate it. △

DSPARMS=*dsparms-parameters*

specifies (for the PDSs) the allocation parameters that are not already specified by the BIGSPACE= and SMLSPACE= parameters. On z/OS when the z/OS UNIX File System area is not being used, the trees are first made with PDSs (by a call to %CPHTREE) and then "copied" to the corresponding directories on UNIX or Windows (by a call to %CMFTPSND). The values for DSPARMS= are a subset of the parameters that are needed to allocate a data set on z/OS by using a SAS FILENAME statement.

This parameter is optional if you are running on z/OS and not using the z/OS UNIX File System area; for other operating environments or for z/OS when you are using the z/OS UNIX File System area, this parameter is ignored.

The supported parameters are:

BLKSIZE=
specifies the block size.

LIKE=
specifies the data set to be used as a model. If LIKE= is not specified, then DSORG=, RECFM=, and LRECL= must be specified. If LIKE= is specified, then DSORG=, RECFM=, and LRECL= are ignored, even if they are specified.

DSORG=
specifies the data set organization.

RECFM=
specifies the record format.

LRECL=
specifies the logical record length.

UNIT=
specifies the unit name. If UNIT= is not specified, then one or more of the following SMS operands must be specified.

DATACLAS=
specifies the SMS data class.

MGMTCLAS=
specifies the SMS management class.

STORCLAS=
specifies the SMS storage class.

VOLSER=
specifies the volume serial number of the unit.

The default is

```
DSPARMS = %QUOTE(dsorg=po, lrecl=6156,
                 blksize=0, recfm=vb)
```

The SAS %QUOTE function protects its contents as a string. That is, it protects the contents from any attempts by the SAS macro compiler to parse the string and then try to evaluate it.

For more information on these parameters, see the z/OS SAS Companion for your current release of SAS.

LISTCLR=YES | NO | QUICKSTART | REMAP
if LISTCLR=YES or LISTCLR=NO or LISTCLR=QUICKSTART, specifies whether %CPHTREE is to clear the "super" tree's list of single trees. If LISTCLR=REMAP, then the parameter specifies that an .htm file is to be created that reports on the contents of the "super" tree and its single trees. This parameter is optional. The default is LISTCLR=YES.

If LISTCLR=YES, then the list is cleared; when this call to %CPHTREE is finished, only one single tree is in the list. This is the single tree that is represented by the SUBLOC= parameter. If LISTCLR=NO, then the list is not cleared. When this call to %CPHTREE finishes, all the single trees that were created since the last clear are on the list and the single tree that is represented by the SUBLOC= parameter is in the list.

If LISTCLR=YES, then the parameters PTITLE1=, PTITLE2=, and PTITLE3= take effect. If LISTCLR=NO, then those parameters are ignored. The setting of the LISTCLR= parameter has no effect on the TITLE1=, TITLE2=, and TITLE3= parameters.

Note: The clear does not delete the "super" tree's existing single trees, but it does remove the "super" tree's ability to know about them and to access them. △

If you use the QuickStart Wizard for multiple data sources, then the setting LISTCLR=QUICKSTART is equivalent to LISTCLR=YES for the first data source that you select in the QuickStart Wizard, and it is equivalent to LISTCLR=NO for any other data sources that you select in the QuickStart Wizard.

If LISTCLR=REMAP, then a file is created that lists the directories in the explorer type of view and, for each directory, lists the reports that currently reside in the directory. The file is named treemap.htm and the value of the SUBLOC= parameter is its pathname. If a user clicks on the Map link that is to the left under the explorer type of view in the left (blue) frame, then this file is displayed in the right (white) frame.

MAXLINES=*number-of-lines*
specifies the maximum number of lines that can be included in the outline, which is specified by the CAT= parameter. This parameter is optional. *The default is 256.*

PTITLE1=*ptitle1-text*
specifies the first heading for the "super" explorer type of view. This heading is displayed in a small, bold, italic font. This parameter takes effect if the LISTCLR= parameter is set to YES or has a value that has an effect of YES. If the LISTCLR= parameter is set to NO or has a value that has an effect of NO, then this parameter is ignored. This parameter is optional.

For WEBSTYLE=GALLERY2, the default is *PTITLE1=SAS IT Management Solutions*. For WEBSTYLE=DYNAMIC, the default is blank.

PTITLE2=*ptitle2-text*
specifies the second heading for the "super" explorer type of view. This heading is displayed in a large, bold, nonitalic font. This parameter takes effect if the LISTCLR= parameter is set to YES or has an effect of YES. If the LISTCLR= parameter is set to NO or has an effect of NO, then this parameter is ignored. This parameter is optional.

For WEBSTYLE=GALLERY2 and WEBSTYLE=DYNAMIC, the default is *PTITLE2=SAS IT Management Solutions Reports*.

PTITLE3=*ptitle3-text*
specifies the third heading for the "super" explorer type of view. This heading is displayed in a much smaller than normal, nonbold, nonitalic font. This parameter takes effect if the LISTCLR= parameter is set to YES or has an effect of YES. If the LISTCLR= parameter is set to NO or has an effect of NO, then this parameter is ignored. This parameter is optional.

For WEBSTYLE=GALLERY2 and WEBSTYLE=DYNAMIC, the default is blank.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

SEPARATR=*separator-character*
specifies the character that separates the second and third, third and fourth, and fourth and fifth fields in each line of the outline in your source entry. This parameter is optional. *The default is SEPARATR=!.*

SMLSPACE=*amount-of-space*
specifies the units of primary space and (optionally) secondary space and directory blocks that are required for each PDS that does not contain reports. This includes the PDS that is named WEB and includes the PDS that is specified by the SUBLOC= parameter. *This parameter is optional if you are running on z/OS and not using the z/OS UNIX File System area; for other operating environments or for z/OS when you are using the z/OS UNIX File System area, this parameter is ignored.*

The PDSs that do not contain reports typically require a few tracks of disk space. For example,

```
SMLPACE=%QUOTE((TRK,(2,7,16)))
```

will result in PDSs that have allocations with the following:

```
SPACE=(TRK,(2,7,16))
```

The SAS %QUOTE function protects its contents as a string. That is, it protects the contents from any attempts by the SAS macro compiler to parse the string and then try to evaluate it. The remaining parameters of the allocation are specified in the DSPARMS= parameter. The default is

```
SMLSPACE=%QUOTE((TRK,(2,5,12)))
```

SUBLOC=*directory-name*
specifies the name of the directory that will be at the top of this single tree. This directory is created under the directory that is specified by the SUPERLOC= parameter. Logically, this directory is above level 0 in the "outline" that is specified by the CAT= parameter.

On z/OS if the z/OS UNIX File System area is not being used, this parameter specifies the right-most qualifier in the name of the PDS that is created at the top of this single tree.

The name must not match the name of any other directory that the call to %CPHTREE will create. That is, the name must not be WEB or the name of any of the directories in the outline. Similarly, if the "super" directory already has directories from other single trees, then the name must not match the name of any of those directories unless you want to overwrite an existing directory.

This parameter is optional. The default is SUBLOC=INFO.

TITLE1=*title1-text*
specifies the first heading for the single-tree explorer type of view. This heading is displayed in a small, bold, italic font. This parameter is not affected by the setting of the LISTCLR= parameter. This parameter is optional.

For WEBSTYLE=GALLERY2, the default is *TITLE1=SAS IT Management Solutions*. For WEBSTYLE=DYNAMIC, the default is blank.

TITLE2=*title2-text*
specifies the second heading for the single-tree explorer type of view. This heading is displayed in a large, bold, nonitalic font. The value of this parameter is also used

to identify the single tree in the "super" explorer type of view. This parameter is not affected by the setting of the LISTCLR= parameter. This parameter is optional.

For WEBSTYLE=GALLERY2 and WEBSTYLE=DYNAMIC, the default is *TITLE2=SAS IT Management Solutions Reports*.

TITLE3=*title3-text*
specifies the third heading for the single-tree explorer type of view. This heading is displayed in a much smaller than normal, nonbold, nonitalic font. This parameter is not affected by the setting of the LISTCLR= parameter. This parameter is optional.

For WEBSTYLE=GALLERY2 and WEBSTYLE=DYNAMIC, the default is blank.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

%CPHTREE Notes

A demonstration "super" tree ships with SAS IT Resource Management. The single trees within it are for the data sources that the QuickStart Wizard supports. Each single tree was created by one call to %CPHTREE by the QuickStart Wizard. Similarly, the reports within the "super" tree are for the data sources that the QuickStart Wizard supports. Each set of reports was created by running the *x*report.sas program that was created by the QuickStart Wizard.

Before you design your own structure, you might want to navigate around in the demonstration "super" tree to get a feel for the kind of structure that is created by the calls to %CPHTREE. To locate the demonstration "super" tree, invoke the SAS IT Resource Management client, and select

OnlineHelp ▶ Other ITRM Documentation ▶ QuickStart Examples

Similarly, before you design your own report program, you might want to look at one or more of the report programs that the QuickStart Wizard generates. To review a report program, run the QuickStart Wizard with any data source (it does not have to be your data source and you do not need to collect that type of data). When you run the wizard, the report program *x*report.sas is created in the *qs*/*cntl* directory (or *.QS.CNTL* PDS) under the PDB that is created by the QuickStart Wizard. When you run the wizard, this directory will contain the programs that are created by the wizard as well as general-purpose files. (You can delete the PDB and everything under it when you are finished looking at the xreport.sas file.)

For more information about using the QuickStart Wizard, see "QuickStart Wizard" in the section "Locating Help" "in the "Chapter 1: Introductions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

For UNIX or Windows environments or for z/OS when the z/OS UNIX File System area is used, the "super" tree represents a complete directory.

On z/OS

- when the z/OS UNIX File System area is not used, the %CPHTREE macro creates single trees and the super tree by using PDSs instead of directories, and it makes members in those PDSs instead of files. The PDSs are created in a location that is based on the value of the SAS system option FILEDEV. Check with the person who installed SAS at your site to determine whether FILEDEV points to a set of temporary volumes. If FILEDEV does point to a set of temporary volumes, then you must move the PDSs to a permanent location.
- after you run %CPHTREE, use the %CMFTPSND macro to create the directories (on UNIX, Windows, or the z/OS UNIX File System) that correspond to the PDSs and to transfer the contents of each PDS to its corresponding directory. For more information, see "%CMFTPSND" on page 37.

%CPHTREE Examples

Example 1

This example creates a tree structure (hierarchical view) of three directories that contain reports that are created by using SAS IT Resource Management report macros. These macros used the following parameters and values: OUTMODE=WEB, WEBSTYLE=GALLERY2. The tree structure is created on the basis of the following directories:

/tmp/myweb/cpu

/tmp/myweb/disk

/tmp/myweb/net

```
%CPHTREE(superloc=/tmp/myweb/, subloc=tree3,
   cat=work.treedefs.tree3.source );
```

The catalog entry WORK.TREEDEFS.TREE3.SOURCE defines the structure of the tree and contains

```
0  CPU Utilization      ! cpu
0  Disk Utilization     ! disk
0  Network Utilization  ! net
```

%CPHTREE will create the HTML files necessary to link the three directories together in the directory /tmp/myweb/tree3. The file /tmp/myweb/welcome.htm will contain a file that points to the top level of the tree.

Example 2

This example is the same as Example 1, except that it also makes a directory for exception reports. The exception reports are generated in the directory /tmp/myweb/ excepts by using the %CPEXCEPT and %CPXHTML macros.

```
%CPHTREE( superloc=/tmp/myweb/, subloc=tree4,
   cat=work.treedefs.tree4.source );
%CPEXCEPT( ruleds=ADMIN.CPXRULE,
   results=ADMIN.PXRULE, );
```

```
%CPXHTML( htmldir=/tmp/myweb/excepts, ruleds=ADMIN.CPXRULE,
   results=ADMIN.PXRULE, );
```

WORK.TREEDEFS.TREE4.SOURCE defines the structure of the tree and contains

```
0  CPU Utilization      !  cpu
0  Disk Utilization     !  disk
0  Network Utilization  !  net
0  Exceptions
    1 Exception Summary ! excepts ! allwelco.htm
    1 Exception Results ! excepts ! excwelco.htm
    1 Exception Reports ! excepts ! welcome.htm
```

Example 3

On z/OS if the z/OS UNIX File System area is not used, the call to the %CPHTREE macro from Example 2 might look like this:

```
%CPHTREE( superloc=TMP.MYWEB, subloc=TREE4,
   cat=work.treedefs.tree4.source,
   bigspace=%QUOTE((CYL,(1,1,10))),
   smlspace=%QUOTE((TRK,(1,1,10))),
   dsparms=%QUOTE(recfm=vb,lrecl=3156,blksize=3160,dsorg=po));
```

Example 4

You can embed site-specific information (such as a company logo) in reports that are organized with the %CPHTREE macro, by using the TITLE1= (or PTITLE1=) parameter. For example, if your company logo is stored in /my/logo/goodlogo.gif, then use the following code:

```
%CPHTREE( superloc=/tmp/cnoabc/qs,
           subloc=pserv,
           cat=pgmlib.jswizcat.jpservtr.source,
           title1=<IMG SRC='/mylogo/goodlogo.gif' height=100 width=100>,
           title2=Server Reports,
           title3=HP OVPA Data);
```

Note that you must use single quotation marks ('...') instead of double quotation marks in the SRC= specification.

Example 5

(For z/OS, when not using the z/OS UNIX File System area) This example creates PDSs that will contain the files that constitute the explorer type of view. The PDSs that make up the tree are found in work.mycat.mytree.source.

```
%CPHTREE(  SUPERLOC=MYUID.PDB.JS,
           SUBLOC=JPSERV,
           BIGSPACE=%QUOTE((CYL,(1,1,10))),
           SMLSPACE=%QUOTE((TRK,(1,1,10))),
           DSPARMS=%QUOTE(recfm=vb,lrecl=3156,blksize=3160,dsorg=po),
           CAT=WORK.MYCAT.MYTREE.SOURCE,
           TITLE2=Server Reports,
           TITLE3=z/OS SMF Data );
```

Example 6

```
/* This SAS program is htree1.sas */

* This program makes two single trees (and the corresponding
explorer-type views) under the same ''super'' directory (or PDS,
if the z/OS UNIX File System area is not used on z/OS).;

* Note: In this sample program, the first single tree is
for reports on HP OVPA data. The second single tree is for
reports on NTSMF exchange data. (There is no real need to do this.
The QuickStart Wizard can make these single trees, and these
"outlines" are copied from the ones used by that Wizard.) ;

* On z/OS, if you do not use the z/OS UNIX File System area,
you would uncomment the setting of the FROM and TO
macro variables in the section immediately below, and fill in
the appropriate value for TO.  You also would uncomment the
call to %CMFTPSND that is after comment #6, and fill in the
appropriate values for the parameters whose names start
with ftp. ;

* See the SAS program htree2.sas for a sample reporting
job that generates a report (based on the data in the
demonstration PDB named pdb-pcs) and directs it into
the first single tree. It also enables you to use the
explorer-type view to see how to select to the report. ;

***;
* Edit these values, if necessary;
***;

* Set PDB to the path and name of any PDB that already exists.
On UNIX and Windows, the PDB is not used, but SAS IT Resource
Management requires a PDB when it starts. On z/OS, when the
z/OS UNIX File System area is not used, the name
of the active PDB is used to build the names for two
intermediate data sets (in the call to the %CMFTPSND
macro). On z/OS, the PDB name must not have a trailing
period. ;
%let PDB=d:\SAS\cpe\pdb-pcs ;

* Set DIR to the path and name of the ''super'' directory
or PDS.  In this sample program, DIR=SUPER, but you are
not required to use that name.;
%let DIR = d:\temp\pdbs\super ;

* Set DIR1 to the name of the directory or PDS that
is to be the top of the first single tree.  This sample
program uses a value of DIR1=mwa, but that name is not
required.;
%let DIR1 = mwa ;

* Set DIR2 to the name of the directory or PDS that
is to be the top of the second single tree.  This sample
```

```
uses a value of DIR2=nexng, but that name is not
required.;
%let DIR2 = nexng ;

/* only for z/OS when z/OS UNIX File System area is not used

* Set FROM to the name of the ''super'' PDS ;
%let FROM = &DIR ;

* Set TO to the name of the ''super'' directory on a
UNIX or Windows host.  The host must have ftp server software;
%let TO = name-of-super-directory ;

*/

***;
* In the .LOG file/LOG window, display these values ;
***;

%put PDB is &PDB ;
%put DIR is &DIR ;
%put DIR1 is &DIR1 ;
%put DIR2 is &DIR2 ;

/* only for z/OS

%put FROM is &FROM ;
%put TO is &TO ;

*/


/*************************************************************
 * #1: Start SAS IT Resource Management and
 *   activate the pdb with ''readonly'' access
 *     Note: On z/OS, you would convert the call to use the z/OS
 *   parameters for %CPSTART.
 *************************************************************/

%cpstart ( pdb = &PDB
         , access = readonly
         , mode = batch
         , _rc = cpstrc
         ) ;

%put CPSTART return code is &cpstrc ;

/* use %cpdeact instead of %cpfailif if running under DMS */
%cpfailif(&cpstrc ne 0, code=999) ;


/*************************************************************
 * #2: In a catalog named TEMP in the library named WORK, make
 * a catalog entry named TEMP1.SOURCE.  Put in it the ''outline''
```

```
 * that represents levels 0, 1, 2, etc. of the first single
 * tree that you want to create.
 * The ''super'' directory (on z/OS, PDS if the z/OS UNIX
 * File System area is not used) and top-of-single
 * -tree directory (on z/OS, the z/OS UNIX File System if
 * the z/OS UNIX File System area is not used) to which
 * this ''outline'' applies will be specified in #3.
 * Reminder: Use blanks, not tabs.
 ************************************************************/

%CPCAT; CARDS4;
0 Overview               ! pservov
0 Utilization
    1 Top Systems
       2 Processor       ! pservtc
       2 Memory          ! pservtm
       2 Storage         ! pservts
       2 Network         ! pservtn
    1 Critical Systems
       2 Processor       ! pservcc
       2 Memory          ! pservcm
       2 Storage         ! pservcs
       2 Network         ! pservcn
       2 Bottlenecks     ! pservcb
0 Service
    1 Availability       ! pservsa
    1 Responsiveness     ! pservsr
    1 Throughput         ! pservst
    1 Exception Summary  ! pservex ! srvwelco.htm ! srvinfo.htm
0 Forecasts
    1 Resource Forecasts ! pserv4r
    1 Service Forecasts  ! pserv4s
0 Exceptions
    1 Exception Summary  ! pservex ! allwelco.htm ! allinfo.htm
    1 Exception Results  ! pservex ! excwelco.htm ! excinfo.htm
    1 Exception Reports  ! pservex
;;;;

%cpcat ( cat=work.temp.temp1.source ) ;


/************************************************************
 * #3: Have %CPHTREE create that single tree and the files
 *   that will represent it in the explorer-type views.
 *     Set LISTCLR=YES to create or refresh the list of single
 *   trees that are known to the ''super'' directory (on z/OS,
 *   PDS).
 *     This call to %CPHTREE then will add this single tree to
 *   the list.
 ************************************************************/

%CPHTREE( superloc=&DIR
        , subloc=&DIR1
        , cat=work.temp.temp1.source
```

```
        , listclr=yes
        , ptitle1=Enterprise-Name
        , title1=Enterprise-Name
        , title2=Reports on HP OVPA Data
        , _rc=cphtrc
        );

%put CPHTREE return code is &cphtrc ;

/* use %cpdeact instead of %cpfailif if running under DMS */
%cpfailif(&cphtrc ne 0, code=999) ;


/************************************************************
 * #4: In a catalog named TEMP in the library named WORK, make
 * a catalog entry named TEMP2.SOURCE.  Put in it the ''outline''
 * that represents levels 0, 1, 2, etc. of the first single
 * tree that you want to create.
 *      The ''super'' directory (on z/OS, PDS if the z/OS UNIX
 * File System area is not used) and top-of-single
 * -tree directory (on z/OS, PDS if the z/OS UNIX File System
 * area is not used) to which this ''outline''
 * applies will be specified in #5.
 *      Reminder: Use blanks, not tabs.
 ************************************************************/

%CPCAT; CARDS4;
0 Overview             ! nexngeo
0 Utilization
    1 Server Health     ! nexngsh
    1 Server Load       ! nexngsl
    1 Server History    ! nexngsi
    1 Server Users      ! nexngsu
    1 Server Queues     ! nexngsq
0 Service
    1 Summary     ! nexngex ! srvwelco.htm ! srvinfo.htm
    1 Availability      ! nexngsa
    1 Throughput        ! nexngst
    1 Responsiveness    ! nexngsr
0 Forecasts
    1 Server Load       ! nexnglc
    1 Server Users      ! nexnguc
0 Exceptions
    1 Summary    ! nexngex ! allwelco.htm ! allinfo.htm
    1 Results    ! nexngex ! excwelco.htm ! excinfo.htm
    1 Reports    ! nexngex
;;;;

%cpcat ( cat=work.temp.temp2.source ) ;


/************************************************************
 * #5: Have %CPHTREE create that single tree and the files
 *   that will represent it in the explorer-type views.
```

```
 *     Set LISTCLR=NO to retain the existing list of single
 *   trees that are known to the ''super'' directory/PDS.
 *     This call to %CPHTREE then will add the name of this
 *    single tree to the list.
 ************************************************************/

%CPHTREE( superloc=&DIR
        , subloc=&DIR2
        , listclr=no
        , cat=work.temp.temp2.source
        , title1=Enterprise-Name
        , title2=Reports on NTSMF Exchange Data
        , _rc=cphtrc
        );

%put CPHTREE return code is &cphtrc ;

/* use %cpdeact instead of %cpfailif if running under DMS */
%cpfailif(&cphtrc ne 0, code=999) ;


/************************************************************
 * #6:  If you are on z/OS and not using the z/OS UNIX File
 *   System area, you must add a call to the %CMFTPSND
 *   macro here, to create the corresponding directories on
 *   a Windows or UNIX and to copy the contents of each PDS to its
 *   corresponding directory.
 *      The call would look similar to the one commented out
 *   below.
 *      Note: The UNIX or Windows host must have ftp server software.
 *      Note: The FTPUSER= parameter must specify an id that
 *   has the authority to create two data sets with the same
 *   prefix as the active PDB.  These data sets are used for
 *   the ftp script and the messages from running the ftp
 *   script.
 *      Note: &FROM is the notation for the value of the FROM
 *   macro variable, which is not to end in a period.  The
 *   second dot is the period between high-level qualifiers.
 ************************************************************/

/* only for z/OS if the z/OS UNIX File System area is not used

%CMFTPSND (&FROM
           &FROM..WEB
           &FROM..MWA
                &FROM..PSERVOV
                &FROM..PSERVTC
                &FROM..PSERVTM
                &FROM..PSERVTS
                &FROM..PSERVTN
                &FROM..PSERVCC
                &FROM..PSERVCM
                &FROM..PSERVCS
                &FROM..PSERVCN
```

```
                &FROM..PSERVCB
                &FROM..PSERVSA
                &FROM..PSERVSR
                &FROM..PSERVST
                &FROM..PSERVEX
                &FROM..PSERV4R
                &FROM..PSERV4S
           &FROM..NEXNG
                &FROM..NEXNGEO
                &FROM..NEXNGSH
                &FROM..NEXNGSL
                &FROM..NEXNGSI
                &FROM..NEXNGSU
                &FROM..NEXNGSQ
                &FROM..NEXNGEX
                &FROM..NEXNGSA
                &FROM..NEXNGST
                &FROM..NEXNGSR
                &FROM..NEXNGLC
                &FROM..NEXNGUC
          , rdir= &TO
          , ftphost=name-of-UNIX-or-PC-host
          , ftpuser=anonymous
          , ftpuserp=youruserid@yourorganization.itstype
          , _rc=cpftrc
          );

%put CPHTREE return code is &cphtrc ;

* use %cpdeact instead of %cpfailif if running under DMS ;
%cpfailif(&cphtrc ne 0, code=999) ;

*/



/*************************************************************
 * #7:  To see the results in terms of directories/PDSs, use
 *   your system's file system to look at the parent
 *   directory/PDS and its related directories/PDSs and files.
 *      You can run the program htree2.sas to see how to load
 *   reports into the trees and how to find the reports in the
 *   explorer-type views.
 *************************************************************/
```

Example 7

```
/* This SAS program is htree2.sas */

* This program generates a report (based on the data in
the demonstration PDB named pdb-pcs) into a tree that
was made by the program htree1.sas.  If you select the Help
link or the Notes link when viewing the report, you will
open a file that was generated automatically.
If you would prefer that the links display files that you
```

```
write, refer to the "Customizing a Report Gallery's Static
Files" section in the "Reporting: Working with Galleries"
chapter in the SAS IT Resource Management User's Guide. ;

***;
* Edit these values, if necessary;
***;

* Set PDB to the path and name of the PDB whose data is to
be the basis of the reports. On z/OS if the z/OS UNIX File System area is
not used, the name of the active
PDB is also used to build the names for two intermediate
data sets (in the call to the %CMFTPSND macro).  On z/OS,
the PDB name must not have a trailing period. ;
%let PDB = d:\SAS\cpe\pdb-pcs ;

* Set DIR to the path and name of the ''super'' directory
(or, on z/OS, PDS if the z/OS UNIX File System area is not used).
 The name is not required to be super. That name is used in
 this sample program to help make the program easy to read. ;

%let DIR = d:\temp\pdbs\super ;

/* only for z/OS if the z/OS UNIX File System area is not used

* Set FROM to the name of the ''super'' PDS ;
%let FROM = &DIR ;

* Set TO to the name of the ''super'' directory on the
UNIX or Windows host.  (You created this ''super'' directory
in the %CMFTPSND call in the htree1.sas program.)
%let TO = name-of-''super''-directory ;

*/


***;
* In the .LOG file/LOG window, display these values ;
***;

%put PDB is &PDB ;
%put DIR is &DIR ;

/* only for z/OS

%put FROM is &FROM ;
%put TO is &TO ;

*/


/*************************************************************
 * #1: Start SAS IT Resource Management and
 *   activate this pdb with readonly access
```

```
 ************************************************************/

%cpstart ( pdb = &PDB
         , access = readonly
         , mode = batch
         , _rc = cpstrc
         ) ;

%put CPSTART return code is &cpstrc ;

/* use %cpdeact instead of %cpfailif if running under DMS */
%cpfailif(&cpstrc ne 0, code=999) ;


/************************************************************
 * #2: Clear the catalog and directory for the exception reports
 ************************************************************/

%CPWEBINI( dir=&DIR\pservcb
         , cat=work.pservcbc
         , list=YES
         , _RC=cpwerc
         );

%put CPWEBINI return code is &cpwerc ;

/* use %cpdeact instead of %cpfailif if running under DMS */
%cpfailif (&cpwerc ne 0, code=999) ;


/************************************************************
 * #3: Construct a macro variable that can be used in the WHERE
 *   clause to subset the data, in this case subsetting the
 *   data to the systems that use the most CPU.
 ************************************************************/

%CPIDTOPN( inlib    = detail
         , inmem    = pcsglb
         , begin    = latest
         , end      = latest
         , ltcutoff = 6:15
         , topnvars = glbcpto
         , macnames = TOPCPU
         , topnclas = machine
         , topNstat = mean
         , topN     = 6
         , _rc      = cpidrc
         );

%put NOTE: CPIDTOPN return code is &cpidrc ;
%cpfailif (&cpidrc ne 0, code=999) ;


/************************************************************
```

```
 * #4: Run a report definition and send the report to the
 *   directory PSERVCB (on z/OS, the PDS whose name ends in
 *   PSERVCB if the z/OS UNIX File System area is not used).
 *   directory/PDS
 *   you created this when you made the first call to %CPHTREE
 *    htree1.sas in the program.
 *     Note: Although this directory/PDS is *logically* under
 *   the top of the single tree's directory/PDS, it is
 *   *physically* directly under the ''super'' directory/PDS.
 *   parent.  (It is the explorer-type views that make the
 *   directories appear to be trees.)
 ************************************************************/

%CPRUNRPT( QPMIXQSK
         , outdesc  = System Queuing
         , gblwhere = &TOPCPU
         , htmldir  = &DIR\pservcb
         , folder   = pgmlib.itsvrpt
         , outmode  = web
         , webstyle = gallery2
         );


/************************************************************
 * #5:  If you are on z/OS and not using the z/OS UNIX File System
 *   area, you must add a call to the %CMFTPSND
 *   macro here, to write the Web-enabled reports to the
 *   corresponding directories that you created with the
 *   %CMFTPSND macro in the htree1.sas program.
 *      The call would look similar to the one commented out
 *   below.
 *      Note: The UNIX or Windows host must have ftp server software.
 *      Note: The FTPUSER= parameter must specify an id that
 *   has the authority to create two data sets with the same
 *   prefix as the active PDB.  These data sets are used for
 *   the ftp script and the messages from running the ftp script.
 *      Note: &FROM is the notation for the value of the FROM
 *   macro variable, which is not to end in a period.  The
 *   second dot is the period between high-level qualifiers.
 ************************************************************/

/* only for z/OS if the z/OS UNIX File System area is not used

%CMFTPSND (&FROM
           &FROM..WEB
           &FROM..MWA
                &FROM..PSERVOV
                &FROM..PSERVTC
                &FROM..PSERVTM
                &FROM..PSERVTS
                &FROM..PSERVTN
                &FROM..PSERVCC
                &FROM..PSERVCM
                &FROM..PSERVCS
```

```
                &FROM..PSERVCN
                &FROM..PSERVCB
                &FROM..PSERVSA
                &FROM..PSERVSR
                &FROM..PSERVST
                &FROM..PSERVEX
                &FROM..PSERV4R
                &FROM..PSERV4S
            &FROM..NEXNG
                &FROM..NEXNGEO
                &FROM..NEXNGSH
                &FROM..NEXNGSL
                &FROM..NEXNGSI
                &FROM..NEXNGSU
                &FROM..NEXNGSQ
                &FROM..NEXNGEX
                &FROM..NEXNGSA
                &FROM..NEXNGST
                &FROM..NEXNGSR
                &FROM..NEXNGLC
                &FROM..NEXNGUC
          , rdir= &TO
          , ftphost=name-of-UNIX-or-PC-host
          , ftpuser=anonymous
          , ftpuserp=youruserid@yourorganization.itstype
          , _rc=cpftrc
          );
%put CPHTREE return code is &cphtrc ;

* use %cpdeact instead of %cpfailif if running under DMS ;
%cpfailif(&cphtrc ne 0, code=999) ;

*/


/************************************************************
 * #6: To see the report, point your Web browser to the
 *   welcome.htm file in the ''super'' directory's WEB directory
 *   or to the welcome.htm file in the ''super'' directory.
 *     Follow this selection path: Reports on HP OVPA
 *   Data -> Utilization -> Critical Systems -> Bottlenecks.
 ************************************************************/
```

%CPIDTOPN

Identifies the n *top (or bottom) contributors to a phenomenon*

%CPIDTOPN Overview

The %CPIDTOPN macro identifies the *n* top (or bottom) contributors to a phenomenon. For example, it can identify the ten users who use the most disk I/O or the five most blocked telephone trunks.

The macro generates one or more expressions that you can use to subset data. For example, in a report definition you can use a WHERE clause that contains one of these expressions and generate reports that contain only the major contributors. The expressions that the macro generates are stored as the values of macro variables whose names you provide as input.

Optionally, the macro generates one or more formats that you can use to separate the major contributors from the others. For example, you can base a formula variable on a format. Then, in a report definition, you can use the formula variable in reports that contain all contributors, but only identify the heavy contributors and collapse all the other contributors into a pool called "Other." The formats that the macro generates are stored in WORK.FORMATS under the names that you provide as input.

%CPIDTOPN Syntax

%CPIDTOPN(
INLIB=*libref*
,**INMEM=***SAS-data-set-name* | *table-name*
,**MACNAMES=***macro-variable-names*
,**TOPN=***integer*
,**TOPNCLAS=***variable-name*
,**TOPNVARS=***variable-name*
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***variable-name*>
<,**END=***SAS-datetime-value*>
<,**FMTNAMES=***format-names*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**_RC=***macro-var-name*>
<,**TOPNSTAT=***statistic*>
<,**WHERE=***where-expression*>);

Details

INLIB=*libref*
in physical terms, specifies the libref of the library that contains the input data set whose data is to be examined. (The libref must be defined by the time that this macro is called.) In logical terms, this parameter specifies the table's level that is to be examined. Typically, the value of the INLIB= parameter is DETAIL (or ADMIN, if the QuickStart Wizard was used), DAY, WEEK, MONTH, or YEAR.
This parameter is required.

INMEM=*SAS-data-set-name* | *table-name*
in physical terms, specifies the name of the data set that contains the observations that are to be examined. In logical terms, specifies the name of the table whose observations are to be examined.
This parameter is required.

MACNAMES=*macro-variable-names*
specifies the name of at least one macro variable. If there is more than one name, then the names must be separated by one or more blanks. A name must start with a letter. It can include letters and numbers, and it can be as long as eight characters. To prevent conflict, if the macro variable exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and given a value. (The number of names that are specified in the MACNAMES= parameter and, optionally, in the FMTNAMES= parameter must match the number of names

that are specified in the TOPNVARS= parameter. The first name of each parameter specifies the first set of input and output, the second name of each parameter specifies the second set, and so on.)

For each set, the name in the MACNAMES= parameter specifies the macro variable that the macro is to use to store the expression that it generates. The value of the macro variable will be an expression of the following form:

```
((BY-variable="value" and
    TOPNCLAS-variable="value") or
 (BY-variable="value" and
    TOPNCLAS-variable="value") or ...
 (BY-variable="value" and
    TOPNCLAS-variable="value") or
 (BY-variable="value" and
    TOPNCLAS-variable="value"))
```

where there are several lines for each group (value of the BY variable) and, within that group, *n* lines for the *n* subgroups (values of the TOPNCLAS variable) that meet the criteria that are specified jointly by the TOPNVARS=, TOPNSTAT=, and TOPN= parameters.

To use the expression in, for example, a report definition, use the function "&" to obtain the value of the macro variable. For instance, if the macro variable's name is HEAVYUSE, then use the following:

```
WHERE &HEAVYUSE
```

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. △

For an example that uses an expression that is generated by %CPIDTOPN, see the examples for the %CPIDTOPN macro.

This parameter is required.

TOPN=*integer*

specifies an integer. The absolute value of the integer indicates the number, *n*, of subgroups that the macro is to identify in the group.

If the integer is positive, then the macro identifies the *n* subgroups with the highest values of the statistic that is specified in the TOPNSTAT= parameter.

If the integer is negative, then the macro identifies the *n* subgroups with the lowest values of the statistic that is specified in the TOPNSTAT= parameter.

This parameter is required.

TOPNCLAS=*variable-name*

specifies the name of the variable that the macro uses to subgroup the observations within each group. There is one subgroup of observations for each value of the variable.

For example, if TOPNCLAS=SHIFT, then the macro subgroups the group's observations by their value of the variable SHIFT. There is one subgroup of observations for each value of the variable SHIFT.

This parameter is required.

TOPNVARS=*variable-name*

specifies the name of at least one variable. If there is more than one name, then the names must be separated by one or more blanks. (The number of names that are specified in the MACNAMES= parameter and the FMTNAMES= parameter must match the number of names that are specified in the TOPNVARS= parameter. The first name of each parameter specifies the first set of input and output, the second name of each parameter specifies the second set, and so on.)

For each set, the name in the TOPNVARS= parameter specifies the variable whose values are to be used to generate a statistic for the subgroup.

This parameter is required.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **dd:mmm:yyyy:hh:mm:ss**, where **dd** is day, **mmm** is month, **yyyy** is year (in two-digit or four-digit notation), **hh** is hour (using a 24-hour clock), **mm** is minute, and **ss** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*variable-name*
specifies the name of the variable that the macro uses to group observations. If the BY= parameter is specified, then there is one group for each value of the variable. If the BY= parameter is not specified, then all observations are in the same group.

For example, if BY=MACHINE, then the macro groups the observations by their value of the variable MACHINE. There is one group of observations for each value of the variable MACHINE.

This parameter is optional.

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FMTNAMES=*format-names*
specifies the name of at least one format. If there is more than one name, then the names must be separated by one or more blanks. A name must start with a letter. It can include letters and numbers, and it can be seven characters long. If the format already exists in WORK.FORMATS, then its definition is overwritten. If the format does not exist in WORK.FORMATS, then it is created there and given a value. (The number of names that are specified in the MACNAMES= parameter and, optionally, the FMTNAMES= parameter must match the number of names that are specified in the TOPNVARS= parameter. The first name of each parameter specifies the first set of input and output, the second name of each parameter specifies the second set of input and output, and so on.)

For each set, the macro generates a format (in WORK.FORMATS) and names the format as specified in the FMTNAMES= parameter. The format is

```
proc format;
 value $format-name
```

```
"BY-variable-value TOPNCLAS-variable-value" = "Y"
"BY-variable-value TOPNCLAS-variable-value" = "Y"
...
"BY-variable-value TOPNCLAS-variable-value" = "Y"
"BY-variable-value TOPNCLAS-variable-value" = "Y"
other = "N" ;
```

where **format-name** is the name that is specified in the FMTNAME= parameter. There are several lines for each group (value of the BY variable) and, within those lines, *n* lines for the *n* subgroups (values of the TOPNCLAS variable) that met the criteria that are specified jointly by the TOPNVARS=, TOPNSTAT=, and TOPN= parameters. "**Y**" indicates that the pair is a major contributor.

One way to use this format in a report definition is to first create a formula variable whose value is based on the format, and then to base the report definition on the formula variable. For example, in the table that is specified by the INMEM= parameter, create a formula variable like this (that is, as defined by %CPDDUTL control statements that are similar to these):

```
set table name = table-name ;
create formula name = formula-name
  kept = yes
  levels = 'detail'
  source = {
    if put(BY-variable-name || '  '
                || TOPNCLAS-variable-name,
     $format-name.) = "Y"
    then formula-name = TOPNCLAS-variable-name ;
    else formula-name = "Other" ;}
  type = character;
build views name = table-name;
```

where each observation's concatenation (of its BY variable's value, with a space, and with its TOPNCLAS variable's value) is mapped (by using the format) to a **Y** or an **N**. If it maps to a **Y**, then the observation is in one of the major contributor group/subgroup pairs, and the value of the formula variable is set to its value of subgroup. Otherwise, the observation is not in one of the major contributor pairs, and the value of the formula variable is set to **Other**. As a result, all the major contributors are specifically identified and all the other contributors are put into a pool that is named **Other**. You can use this formula variable in a report definition to label the major contributors and to see the proportion of the contributions that they represent.

Note: If you report on a statistic that is based on the formula variable, then the report's interpretation must be sensitive to the statistic. Thus, you might want to try several statistics, for example, SUM and MEAN, to see which one makes the report easier to interpret correctly. Also, for accurate interpretation by users of the report, identify the statistic in the report (for instance, in a title line or footnote line). △

For an example that shows how to use a formula variable that is based on a format that is generated by %CPIDTOPN, see Example 3 for the %CPIDTOPN macro.

This parameter is optional. If this parameter is not specified, then the macro does not generate a format.

LTCUTOFF=*time-of-cutoff*

specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the

specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

_RC=*macro-var-name*

specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

TOPNSTAT=*statistic*
specifies the statistic that is to be generated for each subgroup, based on the subgroup's values of the TOPNVARS= variable.

This statistic can be any statistic that is in the list of statistics that are valid for the SUMMARY procedure (PROC SUMMARY), and that list is based on the list of statistics that are valid for the MEANS procedure (PROC MEANS). Typically, one of the following statistics is specified:

N
specifies the number of observations in the subgroup that have nonmissing values for the variable.

NMISS
specifies the number of observations in the subgroup that have missing values for the variable.

MAX
specifies the maximum value of the variable.

MEAN
specifies the mean of the values of the variable (SUM/N).

MIN
specifies the minimum value of the variable.

RANGE
specifies the range of the values of the variable (MAX-MIN).

SUM
specifies the sum of the nonmissing values of the variable.

The other statistic keywords are: SUMWGT, CSS, USS, VAR, STD, STDERR, CV, SKEWNESS, KURTOSIS, T, PRT. For information on these statistics, see "The MEANS Procedure" in the *SAS Procedures Guide* that corresponds to your current version of SAS.

This parameter is optional. If TOPNSTAT= is not specified, then the SUM statistic is used.

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

%CPIDTOPN Notes

Some of the input parameters specify which observations are to be examined. Start with the observations in INLIB.INMEM. Optionally, obtain a subset of the observations. One way to subset is to use the WHERE= parameter. Another way to subset is to use the BEGIN=, END=, and optionally LTCUTOFF= parameters. (If you specify both kinds of subsetting, then the observations that are obtained satisfy both criteria.)

The remainder of the input parameters specify which variable's values will be examined and where the results will be stored. The value of the TOPNVARS= parameter specifies the variable whose values will be examined, and the value of the MACNAMES= and, optionally, FMTNAMES= parameters specify where the results will be stored. The expression is stored in the variable named in MACNAMES= and, optionally, the format is stored as the format named in FMTNAMES=. If more than one expression (and format) will be generated, then these three parameters have multiple values. The first input/output set is based on the first value of each of these parameters, the second set is based on the second value of each of these parameters, and so on.

For each set, the observations (after subsetting, if subsetting is specified) are grouped by the variable that is specified in the BY= parameter. (If the BY= parameter is not used, then all the observations that remain after subsetting are treated as one group.) Within each group, the observations are subgrouped by the values of the variable that is specified in the TOPNCLAS= parameter. For each subgroup, the statistic that is specified in the TOPNSTAT= parameter is calculated for the values of the variable that is specified in the TOPNVAR= parameter. The sign of the TOPN= parameter specifies whether the largest or smallest contributors (subgroups) will be identified. The absolute value of the TOPN= parameter specifies how many contributors (subgroups) will be identified.

The expression and, optionally, the format identify those contributors as pairs of BY= and TOPNCLAS= values.

%CPIDTOPN Examples

Example 1: Running a CPU Utilization Report and a Disk Utilization Report

```
*_____________________________________________________;
* This example runs a CPU utilization report on the
* LATEST day of data in DETAIL.PCSGLB for the top 10
* CPU users of the day and a disk utilization report
* for the top 10 disk using systems that identifies
* which users were responsible for the disk I/O.
* Machine whose names start with "test" are ignored.
*_____________________________________________________;
%CPIDTOPN(  inlib=DETAIL,
            inmem=PCSGLB,
            topNvars=GLBCPTO GLBDKTO,
            macnames=CPUSYS  DISKSYS,
            topNclas=MACHINE,
            topNstat=mean,
            topn=10,
            begin=LATEST-1,
            end=LATEST,
            ltcutoff=6:30,
            where=machine ne: "test" );

* Create plot of each of the top CPU using machines;
%CPPLOT1(
         PCSGLB
        ,GLBCPTO,
        ,WHERE=&CPUSYS
        ,BY=MACHINE
        ,REDLVL=DETAIL
        ,OUTDESC=CPU Utilization
        ,BEGIN=Latest
        ,END=Latest
        ,TABTYPE=INTERVAL
        ,LABELS=(MACHINE="Machine"
                 GLBCPTO="CPU Util")
         );

* Now create a format that identifies the 11 heaviest
* users of disk I/O on each of the 10 machines with
* the highest disk utilization:

%CPIDTOPN (  inlib=detail,
             inmem=glbcpto,
             topNvars=prodskis,
             macnames=DSKUSER,
             fmtnames=DSKUSER,
             topNclas=prounam,
             by=machine,
             where=&DISKSYS,
             topNstat=sum,
             topN=11,
```

```
             begin=latest-1,
             end=latest-1 );


* Define a formula variable to use the new format;
%CPCAT;  cards4;
set table name=glbcpto;
     create formula name=DSKUSER
            levels = 'DETAIL DAY WEEK MONTH YEAR'
            source = {
                    * Avoid warning during migration;
                     length prounam $16.;
                     length machine $32.;
                     if put(trim(machine) || " " ||
                           prounam,$DSKUSER.)="Y"
                           then DSKUSER = prounam;
                      else DSKUSER = "~Others";
                     }
            type   = character
            length = 32
            label  = 'User'
            kept   = YES
            description = 'Name of a top disk user'
            ;
 ;;;;
%CPCAT( cat=work.temp.temp.source);
%CPDDUTL(entrynam=work.temp.temp.source);

* Produce a stacked vertical bar chart report using
* the new formula variable that will show which users
* are responsible for the disk I/O on the 10 heaviest
* disk using systems. If there are more than 11 users
* active on each of these systems, the chart will lump
* the less active users into a single category called
* "~Other".  Limiting the number of users displayed
* keeps the color bands on the vertical bars from
* being too numerous for a readable legend. ;

%CPCHART(
          PCSPRO
         ,HOUR,
         ,PRODSKIS,
         ,BY=MACHINE
         ,REDLVL=DAY
         ,BEGIN=Latest
         ,END=Latest
         ,TABTYPE=INTERVAL
         ,OUTDESC=User Disk I/O
         ,STAT=SUM
         ,SUBGROUP=DSKUSER
         ,WHERE=&DSKUSR
         ,ORDER=Midpoint
         ,type=VBAR
         ,YVAL=40
```

```
,ZERO=YES
,LABELS=(MACHINE="Machine"
         TOPUSER="User"
         PRODSKIS="Disk I/Os"
         HOUR="Hour")
);
```

Example 2: Running a CPU Utilization Report

This example runs a CPU utilization report on the LATEST-1 day, by using the data in DETAIL.PCSGLB for the top ten CPU users of the day. It also runs a disk utilization report for the top ten disk users of the LATEST-1 day.

Machines whose names start with "test" are ignored for both reports.

```
%CPIDTOPN (inlib=DETAIL,
           inmem=PSCGLB,
           topNvars=GLBCPTO GLBDKTO,
           topNclas=MACHINE,
           topNstat=mean,
           topN=10,
           begin=LATEST-1,
           end=LATEST-1,
           where=machine not ne: "test",
           macnames=MYTOPCP MYTOPDK );
```

The following establishes a subsetting expression for the upcoming CPU utilization report:

```
%CPRUNRPT (QPCPUUX4, folder=gmlib.itsvrpt,
           gblwhere=(%MYTOPCP(glbcpto)));
```

The following establishes a subsetting expression for the upcoming disk utilization report:

```
%CPRUNRPT (QPDSKUSK, folder=pgmlib.itsvrpt,
           gblwhere=(%MYTOPDK(glbdkto)));
```

Example 3: Using an Expression Generated by %CPIDTOPN

This SAS program activates the phone PDB and examines data in the detail level of the rtrunk table. It generates an intermediate report in one Web directory and six versions of a final report in the other Web directory. When you select the welcome.htm files in those directories, you will see the reports. The intermediate report is not necessary, but it might help you understand the variations of the final report.

The final reports are six variations of the same report:

- two reports that do not use the expression or format (one displays means, and one displays sums)
- two reports that use the expression (one displays means, and one displays sums)
- two reports that use the format (one displays means, and one displays sums).

The final reports illustrate how to use the expression and format. They also illustrate how the choice of statistic can affect the report.

Edit these values, if necessary:

- Set PDB to the path and name of the demonstration PDB with phone data. Because you will be changing the PDB by writing to it, you might prefer to make a copy of the PDB and use the copy.

```
%let PDB = d:\temp\pdbs\pdb-pbx ;
```

- Set WEBCATV to the library and catalog name of the intermediate catalog that is to be used to prepare a report that you can use in order to verify the calculations.

```
%let WEBCATV = admin.webcat_t ;
```

- Create a different directory and set WEBDIRV to the path and name of the directory. A welcome.htm file will be generated in this directory and will display a report that you might want to use in order to verify the calculations. To see the report on the Web, run the version of SAS that shipped with SAS IT Resource Management 2.2 or a later release, or download the extra HTML Formatting Tools that were shipped with that version of SAS.

```
%let WEBDIRV = d:\temp\pdbs\webdir_t ;
```

- Set WEBCAT to the library and catalog name of the intermediate catalog that is to be used to prepare three sample reports for the Web.

```
%let WEBCAT = admin.webcat_n ;
```

- Create a directory and set WEBDIR to the path and name of the directory. A welcome.htm file will be generated in this directory. Select the file in order to display the three sample reports.

```
%let WEBDIR = d:\temp\pdbs\webdir_n ;
```

- In the .LOG file/LOG window, display these values:

```
%put NOTE: PDB is &PDB ;
%put NOTE: WEBCATV is &WEBCATV ;
%put NOTE: WEBDIRV is &WEBDIRV ;
%put NOTE: WEBCAT is &WEBCAT ;
%put NOTE: WEBDIR is &WEBDIR ;
```

1 Start SAS IT Resource Management and activate this PDB with write access:

```
%cpstart ( pdb = $PDB
   , access = write
   , mode = batch
   , _rc = cpstrc ) ;
%put CPSTART return code is &cpstrc ;
%cpfailif(&cpstrc ne 0, code=999) ;
```

2 For each shift, get the names of the two groups that have the highest mean of blockage. The variable GOS measures blockage.

Have the macro store the expression in MCEXPR (major contributor expression) and store the format in MCFRMT (major contributor format).

```
%cpidtopn ( inlib = detail
          , inmem = rtrunk
          , by = shift
          , topnclas = grpname
          , topnvars = gos
          , topnstat = mean
          , topn = 2
          , macnames = MCEXPR
          , fmtnames = MCFRMT
          , _rc = cpidrc  );
%put CPIDTOPN return code is &cpidrc ;
%cpfailif(&cpidrc ne 0, code=999) ;
```

3 In the .LOG file/LOG window, display the expression that was generated:

```
 %put MCEXPR is &MCEXPR ;
```

4 In the .LST file/OUTPUT window, display the mapping that is described in the format that was generated:

```
options linesize=80 ;
%put MCFRMT is ;
proc format fmtlib lib=work.formats;
     select $mcfrmt ; run ;
```

5 Clear the catalog and directory that will be used to display an intermediate report that you might want to use in order to understand the variations of the final report:

```
%CPWEBINI (cat = &WEBCATV
         , dir = &WEBDIRV
         , list = YES
         , _rc = cpwerc );
%put CPWEBINI return code is &cpwerc ;
%cpfailif(&cpwerc ne 0, code=999) ;
```

6 Generate an intermediate report that you can use to verify the group/subgroup pairs that were identified as the major contributors.

To see the report, select the welcome.htm file in the directory that is represented by WEBDIRV, and look in report group "TGrpStatsByShift."

```
TITLE1 'Trunk Group Statistics (Mean and Sum)
        by Shift' ;
FOOTNOTE1 '1.00 represents 100%. GOS = blockage = '
          'calls that were temporarily or'
          ' permanently delayed.' ;
%CPTABRPT(RTRUNK
        , GOS,,
        , TYPE=TYPE2
        , CL_NAM=GRPNAME
        , BY=SHIFT
        , REDLVL=DETAIL
        , TABTYPE=INTERVAL
        , STAT=MEAN SUM
        , STATLAB="Arithmetic Average"
        , STATFMT=.
        , MISS=YES
        , PRTMISS=NO
        , SEPS=YES
        , FORMATS=(GOS= 8.3)
        , LABELS=(SHIFT="Operations shift"
                  %nrstr(GOS="% All Trunks Busy
                     <Blockage>")
                  GRPNAME="Trunk Group Name")
        , OUTMODE=WEB
        , PALETTE=PGMLIB.PALETTE.WEB
        , HTMLDIR=&WEBDIRV
        , WEBSTYLE=GALLERY2
        , OUTLOC=&WEBCATV
        , OUTNAME=VERRPT
        , OUTDESC=TGrpStatsByShift  );
```

7 In the .LST file/OUTPUT window, display the format ($TRUNK) that is assigned to the GRPNAME variable. The format maps the abbreviations for group names to their full names.

You will need the mapping to connect the group names (long group names) in the intermediate report to the group names (abbreviations) in the expression and format. This format has nothing to do with the macro. It is a format that is shipped in the phone PDB's data dictionary that makes it easier to read reports about trunk groups.

```
%put TRUNK is ;
proc format fmtlib lib=dictlib.cpfmts;
     select $trunk ;
run ;
```

8 Clear the catalog and directory to which the variations of the final report will be directed:

```
%CPWEBINI (cat = &WEBCAT
         , dir = &WEBDIR
         , list = YES
         , _rc = cpwerc     );
%put CPWEBINI return code is &cpwerc ;
%cpfailif(&cpwerc ne 0, code=999) ;
```

9 Generate a report that does *not* use the expression or format.

To see the report, select welcome.htm in the directory that is represented by WEBDIR, and then select the report that is titled ORIGINAL MEAN (original report, displaying means) or ORIGINAL SUM (original report, displaying sums). Notice that the contributors are not pooled. Each individual contributor (trunk group) is represented separately. If you had many contributors, then the report would be difficult to read and you might run out of space for the legend.

```
TITLE1 'ORIGINAL MEANS' ;
FOOTNOTE1 'Each sub-division is represented by '
          'the mean of its observations.' ;
%CPCHART(RTRUNK
       , SHIFT,
       , GOS,
       , REDLVL=DETAIL
       , TYPE=HBAR
       , SUBGROUP=GRPNAME
       , STAT=MEAN
       , OUTMODE=WEB
       , PALETTE=PGMLIB.PALETTE.WEB
       , HTMLDIR=&WEBDIR
       , WEBSTYLE=GALLERY2
       , OUTLOC=&WEBCAT
       , OUTDESC=Administration    );
TITLE1 'ORIGINAL SUMS' ;
FOOTNOTE1 'Each sub-division is represented by '
          'the sum of its observations.' ;

%CPCHART(RTRUNK
       , SHIFT,
       , GOS,
       , REDLVL=DETAIL
       , TYPE=HBAR
```

```
       , SUBGROUP=GRPNAME
       , STAT=SUM
       , OUTMODE=WEB
       , PALETTE=PGMLIB.PALETTE.WEB
       , HTMLDIR=&WEBDIR
       , WEBSTYLE=GALLERY2
       , OUTLOC=&WEBCAT
       , OUTDESC=Administration  );
```

10 Generate a report that uses the expression for subsetting.

To see the report, select welcome.htm in the directory that is represented by WEBDIR, and then select the report that is titled EXPRESSION MEANS (a report that uses expression and displaying means) or EXPRESSION SUMS (a report that uses expression and displaying sums). Notice that the report contains information about only the major contributors. Each major contributor is represented separately. The minor contributors are not present.

Thus, as long as the number of major contributors is relatively small, the report is easy to read. The number of minor contributors does not affect the readability of the report or the size of the legend.

```
TITLE1 'EXPRESSION MEANS' ;
FOOTNOTE1 'Each sub-division is represented by '
         'the mean of its observations.' ;
%CPCHART(RTRUNK
        , SHIFT,
        , GOS,
        , REDLVL=DETAIL
        , TYPE=HBAR
        , SUBGROUP=GRPNAME
        , STAT=MEAN
        , WHERE=&MCEXPR
        , OUTMODE=WEB
        , PALETTE=PGMLIB.PALETTE.WEB
        , HTMLDIR=&WEBDIR
        , WEBSTYLE=GALLERY2
        , OUTLOC=&WEBCAT
        , OUTDESC=Administration    );
TITLE1 'EXPRESSION SUMS' ;
FOOTNOTE1 'Each sub-division is represented by'
          ' the sum of its observations.' ;
%CPCHART(RTRUNK
       , SHIFT,
       , GOS,
       , REDLVL=DETAIL
       , TYPE=HBAR
       , SUBGROUP=GRPNAME
       , STAT=SUM
       , WHERE=&MCEXPR
       , OUTMODE=WEB
       , PALETTE=PGMLIB.PALETTE.WEB
       , HTMLDIR=&WEBDIR
       , WEBSTYLE=GALLERY2
       , OUTLOC=&WEBCAT
       , OUTDESC=Administration     );
```

The demonstration PDB already has the format $trunk; it specifies that values of the GRPNAME variable are to be displayed by using it. Because the MCFVAR variable has the same values, the $trunk format is also assigned to MCFVAR. (Again, this is for convenience in reading the reports and has nothing to do with the macro.)

```
%CPCAT; CARDS4;
 set table name=rtrunk;
 delete formula name=mcfvar noerror ;
 create formula name=mcfvar
        kept=yes
        levels='detail'
        source={if put(shift||' '||grpname,
                $mcfrmt.)="Y" then
                  mcfvar = grpname;
              else
                  mcfvar = "Other";}
        format = $trunk.
        type=character;
build views name = rtrunk ;
;;;;
%cpcat ( cat=work.temp.mcfvar.source ) ;

%cpddutl (entrynam = work.temp.mcfvar.source
        , list = Y   );
```

11 In the RTRUNK table, create a formula variable MCFVAR (major contributors formula variable). Base the formula variable on the format MCFRMT.

12 Generate a report that uses the formula variable MCFVAR, which is based on the format MCFRMT.

To see the report, select welcome.htm in the directory that is represented by WEBDIR, and then select the report that is titled FORMULA MEANS (a report that uses formula and displaying means) or FORMULA SUMS (a report that uses formula and displaying sums).

Notice that the report contains information about the major contributors and also about the minor contributors. Each major contributor is represented separately. All the minor contributors are pooled and represented together. Thus, as long as the number of major contributors is relatively small, the report is easy to read. The number of minor contributors does not affect the readability of the report or the size of the legend.

```
TITLE1 'FORMULA MEANS' ;
FOOTNOTE1 'Each sub-division is represented by '
          'the mean of its observations.' ;
%CPCHART (RTRUNK
        , SHIFT,
        , GOS,
        , REDLVL=DETAIL
        , TYPE=HBAR
        , SUBGROUP=MCFVAR
        , STAT=MEAN
        , OUTMODE=WEB
        , PALETTE=PGMLIB.PALETTE.WEB
        , HTMLDIR=&WEBDIR
        , WEBSTYLE=GALLERY2
        , OUTLOC=&WEBCAT
```

```
        , OUTDESC=Administration  );
TITLE1 'FORMULA SUMS' ;
FOOTNOTE1 'Each sub-division is represented '
          'by the sum of its observations.' ;
%CPCHART (RTRUNK
        , SHIFT,
        , GOS,
        , REDLVL=DETAIL
        , TYPE=HBAR
        , SUBGROUP=MCFVAR
        , STAT=SUM
        , OUTMODE=WEB
        , PALETTE=PGMLIB.PALETTE.WEB
        , HTMLDIR=&WEBDIR
        , WEBSTYLE=GALLERY2
        , OUTLOC=&WEBCAT
        , OUTDESC=Administration );
```

Example 4: Using FMTNAMES to Subset Formula Variable Definitions

Using the format feature for subsetting is especially useful in SAS IT Resource Management formula variable definitions that can be used to produce plots that explicitly name the busiest machines and group all other values into an "Other" category. For example, this sequence of %CPDDUTL statements defines a formula variable that has the value of MACHINE when the machine is one of those with the busiest disks and has the name "~Others" otherwise:

```
create formula name=DISKHOG
   levels = 'DETAIL DAY WEEK MONTH YEAR'
   source = {
   /*_____________________________________________________________
      * Formula variable DISKHOG has the value of
      * MACHINE if that disk is identified as a busy
      * disk by the SAS informat $DSKHOGS; otherwise it
      * is the name ''~Others''.
      * Format DSKHOGS can be created using the
      * macro produced by the CPIDTOPN macro and
      * setting FMTNAMES=DSKHOGS. Thus, this variable
      * can be used to restrict the scope of a report
      * to only the most active disks on a specific
      * subset of the data (this is necessary to avoid
      * a cluttered legend in stacked bar charts).
      * For example:
      *    %CPIDTOPN( inlib    = detail,
      *               inmem    = pcsglb,
      *               topnvars = glbdkto,
      *               macnames = DSKHOGS,
      *               fmtnames = DSKHOGS,
      *               topNstat = mean,
      *               topnclas = machine,
      *               topN     = 3,
      *               begin    = latest - 1,
      *               end      = latest,
      *               _rc      = mwrc );
      *_____________________________________________________________*/
```

```
length machine $32.;  * Avoid warning during
                      * formula creation;
  if put (machine,$DSKHOGS.) eq "Y"
     then diskhog=machine ;
     else diskhog="~Others" ;    }
type   = character
  length = 32
  label  = 'Disk'
  kept   = YES
  description =
     'Name of one of the busiest disk using machines.' ;
```

Like MACNAMES=, FMTNAMES= can list multiple names that are created independently within a single invocation of %CPIDTOPN and can be used separately later on.

%CPMANRPT

Deletes older reports in a SAS IT Resource Management Web gallery, based on specified criteria

%CPMANRPT Overview

If you are using SAS IT Resource Management to create reports that you can display by using a Web browser, you might have large galleries of Web reports that you need to manage. %CPMANRPT enables you to have a specified group of reports and to add only the newest data to them every day. This is already possible with SAS IT Resource Management, but the gallery keeps growing. Therefore, if you have a gallery to which you add reports on a regular basis (using a permanent SAS catalog), you might want to delete or age out older reports to make room for newer ones. %CPMANRPT enables you to delete reports in a SAS IT Resource Management Web gallery according to criteria such as name, values of BY variables, or specific dates or ranges of dates.

To delete reports by using %CPMANRPT, the SAS catalog that you specify in the OUTLOC= parameter must be a permanent SAS catalog. If your report gallery is created from scratch with each report job, or if the catalog that is specified by OUTLOC= is stored in the WORK library, then you do not need to (and cannot) manage the reports by invoking %CPMANRPT.

%CPMANRPT Syntax

%CPMANRPT(
<*reports*>
,**HTMLDIR=***directory-name* | *PDS-name*
,**KEEP=**(AGE=*days*<BEFORE=*date*>) |
DELETE=ALL | (DATE=*beginning_date* < - *ending_date*>) |
(NAME=*list-of-names*)
,**OUTLOC=***SAS-catalog-name*
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**NODATE=**KEEP | DELETE>
<,**RPTDATE=**(DATEVAR=*date-variable* | DTVAR=*datetime-variable* |
RUNDATE)>
<,**LIST=**YES | NO>

<,**_RC=***macro-var-name*>
<,**WEBSTYLE=***style*>) ;

Details

reports
specifies the name(s) of one or more reports that you want to remove. The report name is used with the age specified and other parameters in order to determine the older reports that you want to age out.

Note: The report definitions themselves are *not* removed. △

If you do not specify this parameter, then by default the macro removes all reports (in the catalog that is specified in the OUTLOC= parameter) that meet the aging criteria. In other words, if any names are given, then the aging criteria apply only to those names. Otherwise, the aging criteria apply to all reports in the catalog.

If you specify more than one name, separate the names with blanks.

HTMLDIR=*directory-name* | *PDS-name*
specifies the complete path and name of the directory or the fully qualified name of the PDS that contains the HTML files that you want to remove. *This parameter is required.* (This is the directory or PDS that was specified in the HTMLDIR= parameter when you originally created the reports.)

Note: The static files are not deleted when reports are aged out. For more information about the static files, see the topic "Customizing a Report Gallery's Static Files" in the chapter "Reporting: Working with Galleries" in the *SAS IT Resource Management User's Guide*. △

KEEP=(AGE=*days*<BEFORE=*date*>)
specifies the reports that you want to keep. *You must specify either the KEEP= parameter or the DELETE= parameter, but not both.*

If you specify the *reports* parameter, then the KEEP= criterion applies only to those specified reports. If you do not specify the *reports* parameter, then the KEEP= criterion applies to all reports in the HTMLDIR= and OUTLOC= locations.

AGE=
specifies the number of days of reports that you want to keep. If you specify the KEEP= parameter, then the AGE= parameter is required.

BEFORE=
specifies the date that should be used to determine the number of days of reports to keep. This parameter is optional. By default, the value that you specify for the AGE= parameter starts with the current date (the date when the SAS program is running) and counts back the number of days that you specify. You can use the BEFORE= parameter to specify a different date from which to start counting. For example, if you specify AGE=10 and then specify a BEFORE= date that is five days earlier than the date on which the program will run, then reports 10 days prior to the BEFORE= date are kept.

The date should be specified in the form **`ddmmmyyyy`** or **`ddmmmyy`**.

Note that the SAS system option YEARCUTOFF might affect the interpretation of a two-digit year. For more information, see the "Using the YEARCUTOFF SAS System Option" section in the "Administration: Working with Data" chapter in the *SAS IT Resource Management User's Guide*.

DELETE= ALL | (DATE=*beginning_date* < - *ending_date*>) | (NAME=*list-of-names*)
specifies the reports that you want to delete. *Specify either the KEEP= parameter or the DELETE= parameter, but not both.*

- *ALL* means that the copies of reports are to be deleted, regardless of their dates.

  If you specify the *reports* parameter, then the DELETE= criterion applies only to those specified reports. If you do not specify the *reports* parameter, then the DELETE criterion applies to all reports in the HTMLDIR= and OUTLOC= locations.
- *DATE*= means that the copies of reports are to be deleted only if the dates (based on the RPTDATE= parameter) match the specified date (if one date is specified) or fall in the specified range (if two dates are specified, separated by a hyphen and optional blanks on either side of the hyphen).

  If you specify the *reports* parameter, then the DELETE= criterion applies only to those specified reports. If you do not specify the *reports* parameter, then the DELETE criterion applies to all reports in the HTMLDIR= and OUTLOC= locations, based on the value of the RPTDATE= parameter.

  To delete all reports that run on a specific date, specify only one date. All reports that run on that date are deleted, based on the value of the RPTDATE= parameter, which identifies how the date value is determined.

  If you specify a beginning date and an ending date, then all reports in that date range (including the beginning and ending dates) are deleted, based on the value of the RPTDATE= parameter.

  The date(s) should be specified in the form **ddmmmyyyy** or **ddmmmyy**. Note that the SAS system option YEARCUTOFF might affect the interpretation of a two-digit year. For more information, see the "Using the YEARCUTOFF SAS System Option" section in the "Administration: Working with Data" chapter in the *SAS IT Resource Management User's Guide*.
- *NAME*= means that the report(s) that is in the list of names is to be deleted. The list of names consists of one or more names that specify the catalog entries whose reports are to be deleted. Use one-level catalog entry names. If you specify more than one name on which the reports are based, separate adjacent names with a comma or with one or more blanks.

  For example, the following means that the report that is based on the entry named _0000005 is to be deleted:

  ```
  DELETE=(NAME=_0000005)
  ```

  When NAME= is specified, the reports parameter is not specified (or, if specified, is ignored).

OUTLOC=*SAS-catalog-name*
specifies the name of the catalog that contains the reports that you want to delete. The value must be specified in the format *libref.catalog_name*. (This is the name that was specified in the OUTLOC= parameter when you originally created the reports.)

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myidir=c:\www\greports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

NODATE=KEEP | DELETE

specifies whether to keep or delete reports when the user-specified date variable is not present in the report definition. This parameter is optional.

Note: This parameter works only when the RPTDATE= parameter is specified with the DATEVAR= option (which specifies a variable that contains the date value) or the DTVAR= option (which specifies a variable that contains the datetime value) for this report. △

KEEP keeps the reports that do not contain the variable that is referenced on the DATEVAR= or DTVAR= option of the RPTDATE= parameter. This is the default.

DELETE deletes the reports that do not contain the variable that is referenced on the DATEVAR= or DTVAR= option of the RPTDATE= parameter.

RPTDATE=(DATEVAR=*date-variable* | DTVAR=*datetime-variable* | RUNDATE)
specifies how the macro should determine the date of the report. This parameter is not required. Valid values are as follows:

DATEVAR=*date-variable*
indicates that you want to use dates that are in your data when determining the date or date range of the DELETE= parameter. The value of the DATEVAR= option is the name of a variable that contains the date value for this report. The variable must be in the BY variables list for this report.

DTVAR=*datetime-variable*
indicates that you want to use datetimes that are in your data when determining the date or date range of the DELETE= parameter. The value of the DTVAR= option is the name of a variable that contains the datetime value for this report. The variable must be in the BY variables list for this report.

RUNDATE
indicates that you want to use the actual date that the report is run. *This is the default value.*

LIST=YES | NO
specifies whether or not to print in the SAS log the list of the files that were deleted. *The default is YES.*

If you specify YES, then the list of files that were deleted is printed in the SAS log. If you specify NO, then this list is not printed in the SAS log.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

%CPMANRPT Notes

%CPMANRPT can manage reports only in a permanent SAS catalog that is populated by %CPRUNRPT, %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC, %CPPRINT, %CPTABRPT, and/or %CPSRCRPT macro calls with OUTMODE=WEB. It cannot manage reports in a catalog that has been populated by %CPXHTML. This means that any follow-up reports related to rules processing cannot be managed by %CPMANRPT.

The static files are not deleted when reports are aged out. For more information about the static files, see the topic "Customizing a Report Gallery's Static Files" in the chapter "Reporting: Working with Galleries" in the *SAS IT Resource Management User's Guide*.

The WEBSTYLE= parameter specifies the style of the gallery that %CPMANRPT rebuilds after it ages out reports (as specified by the KEEP= parameter or DELETE= parameter).

%CPMANRPT Examples

Example 1

This example deletes any report for a date earlier than 31 days before the current date and assumes that all reports in the catalog **sasuser.webage** have DATE as a BY variable. If any report does not have DATE as a BY variable, then an ERROR message will be printed in the SAS log and no reports will be deleted.

```
%cpmanrpt(htmldir=c:\webtest\aging,
          outloc=sasuser.webage,
          keep=(age=31),
          rptdate=(datevar=date),
          list=YES );
```

Example 2

This example deletes reports that are generated by the saved report definition REPORT1 that were run earlier than 28 days before the current date. Because the RPTDATE= parameter is not specified, the macro defaults to using the date that the report was generated. Any reports in the catalog that have a different name are not deleted.

```
%cpmanrpt(REPORT1,
          htmldir=c:\webtest\aging,
          outloc=sasuser.webage,
          keep=(age=28),
          list=YES );
```

%CPPLOT1

Generates a plot with one Y axis

%CPPLOT1 Overview

The %CPPLOT1 macro plots a maximum of 15 analysis (Y) variables against an X variable. The variable that you specify for the X variable corresponds to the X axis. If you specify a class or grouping variable (CL_NAM=), then you can specify only one analysis (Y) variable. This macro supports a number of plot types, which are defined in the description of the TYPE= parameter.

Only one axis is used to display all analysis (Y) variable values; therefore, multiple analysis (Y) variables should be of the same unit of measurement.

For more information on the underlying procedure, refer to the GPLOT procedure in the SAS/GRAPH documentation for your current release of SAS.

%CPPLOT1 Syntax

%CPPLOT1(
dataset
,*variable-label-pairs*
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***BY-variable-list*>
<,**CL_LAB=***label*> (obsolete; use LABELS=)
<,**CL_NAM=***variable-name*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**HAXIS=***value-list* | AXIS*n* | contents of SAS AXIS*n* statement>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>

<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**PERCENTS=**>
<,**PERIOD=***time-interval*>
<,**PROCOPT=***string*>
<,**REDLVL=***PDB-level*>
<,**SKIPMISS=**YES | NO>
<,**SMALLDEV=***device-driver*>
<,**STAT=***statistic*>
<,**STMTOPT=***string*>
<,**TABTYPE=**INTERVAL | EVENT>
<,**TYPE=***plot-type*>
<,**UNIFORM=**YES | NO>
<,**VAXIS=***value-list* | AXIS*n* | Low to High>
<,**VREF=***value-list*>
<,**VZERO=**NO | YES>
<,**WEBSTYLE=***style*>
<,**WEIGHT=***weight-variable*>
<,**WHERE=***where-expression*>
<,**XLAB=***label*> (obsolete; use LABELS=)
<,**XVAR=***variable*>);

Details

dataset
specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

variable-label-pairs
specifies a list of pairs. (For %CPCHART, you can use only one pair.) Each pair consists of the name of one analysis variable and the label for that analysis variable. Each label has a maximum length of 16 characters.

The maximum number of pairs that you can specify is 15, depending on the type of report that you are running. *For a detailed explanation of the variables that are displayed on each axis and the maximum number of variables that can be specified for the report, refer to the "Overview" section of the macro description.*

If you specify multiple analysis variables, then the unit of measure for all of the analysis variables must be the same and the range of values should be similar.

The analysis variable is typically displayed on the left (Y) axis. However, certain combinations of parameters might affect the axis on which the analysis variable is displayed or the number of variables that can be specified, depending on the report macro that you use.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of a label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in a label.

You are required to specify at least one variable in this positional parameter. However, you are not required to specify labels when you use this parameter. In fact, using the LABELS= parameter to specify the labels is preferred. Because this is a positional parameter, you must use a comma as a placeholder for the label when you specify more than one variable but do not specify labels. For example, to specify three variables and no labels, you could specify **`var1,,var2,,var3,KEYWORD=...`**. In this example, you have specified variable 1 but no label for variable 1, variable 2 but no label for variable 2, and variable 3 but no label for variable 3. You then specify any keyword parameters, such as the LABELS= parameter, that you want to use in the report definition.

You can specify the labels by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the LABELS= parameter, then any labels that are specified by using this parameter are ignored. If labels are not specified by using this parameter and not specified by using the LABELS= parameter, then the variables' labels in the data dictionary are used.

BEGIN=*SAS-datetime-value*

specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **`dd:mmm:yyyy:hh:mm:ss`**, where **`dd`** is day, **`mmm`** is month, **`yyyy`** is year (in two-digit or four-digit notation), **`hh`** is hour (using a 24-hour clock), **`mm`** is minute, and **`ss`** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **`dd:mmm:yyyy`** AT **`hh:mm:ss`**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **`hh:mm:ss`**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **`hh:mm:ss`** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*BY-variable-list*

lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

CL_LAB=*label (obsolete; use LABELS=)*
specifies the label for the class variable. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

CL_NAM=*variable-name*
specifies the name of the class variable for the report. For each unique value of the class variable, the macro creates a separate portion or group on the graph.

Compare this type of variable to the variable specified by the macro's BY=parameter. The BY variable creates separate graphs for each unique value of the variable. For example, the following combination of parameters produces one report or plot for each unique value of the BY variable MACHINE:

```
cl_nam=diskid,
by=machine
```

Each of the graphs for MACHINE includes all possible values of DISKID, the class variable. If you are producing a plot, then each unique value of DISKID is represented by a different color or plot symbol. If you are producing a chart, then each value of DISKID is represented by a different column or row.

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*
lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the

format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

HAXIS=AXIS*n* | *value-list* | contents of SAS AXISn statement
specifies the value for the major tick marks on the X axis (the horizontal axis), or an axis number that has been previously defined in a palette, or the contents of a SAS AXIS statement without the word AXIS keyword or the ending semicolon. You can specify the value of the HAXIS= parameter in one of the following ways:

- the axis number, as in *axisn*, where *n* represents the axis number that you have already defined in the palette that you are currently using for this report
- a list of values for the axis
- the contents of a SAS AXISn statement (without the word AXIS at the beginning or the punctuation mark ";" at the end). In other words, anything that you can specify in an AXISn statement you can specify here.

If you specify an axis number, then also specify the PALETTE= parameter in order to indicate the name of the palette that contains this axis.

These examples illustrate ways that you can specify the list of values (*value-list*):

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

Note: The values for HAXIS= must be specified as actual values, not formatted values. Thus, the tick marks for time values must be either a count of seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

The values for the HAXIS= parameter should reflect the range of values from your data that you want to represent in your report. Data that does not fall within the range of values set in the HAXIS= parameter will not be represented in the report. You will receive a warning message if all of your data does not fall within the range of values in the HAXIS= value list, but the report will continue processing.

For more information about the *value-list*, see the ORDER= parameter in the AXIS Statement section of the SAS/GRAPH documentation for your current version of SAS.

Similarly, for more information about the contents of the AXIS statement, see the AXIS Statement section of the SAS/GRAPH documentation for your current version of SAS.

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhdir=c:\www\hreports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR=

parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR=

parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*
specifies the location (URL address) that is used in your HTML output to locate your report images. In **welcome.htm** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)
specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use

matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).

- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)

- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

  Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

```
options
   appletloc="..\path";
```

where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*

specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

OUTDESC=*output-description*

if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*

for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT*: WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if*

you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)

- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename | entry-name*

for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT*: G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management

client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

PALETTE=*palette-name*

specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **`sasuser.palette.win`** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PERCENTS=
is not currently used.

PERIOD=*time-interval*
is the summarization period for the report. *The default is ASIS*. For values other than ASIS, the values for the time period within a class or category are combined (as specified in the STAT= parameter) to form a single point on a plot, a bar on a chart, or a row or column in a table.

The following list provides the period name, the effect that it has on the report, and an example of how to specify the period.

- ASIS - leaves datetime in its present form. Example: 09Jun2003:14:37:25
- 15MIN - transforms the time to the first minute of the 15-minute period in the hour and retains the date. Example: 09Jun2003:14:30
- HOUR - transforms the time to the first minute of the hour and retains the date. Example: 09Jun2003:14:00
- 24HOUR - transforms the time to its hour component and omits the date. Range: 0–23. Example: 14
- DATE - omits the time and retains the date. Example: 09Jun2003
- WHOLEDAY - obsolete; use DATE.
- WEEKDAY - transforms the day to the day of the week (where 1=Sunday, 2=Monday, and so on) and omits the month, year, and time. Range: 1–7, which displays as Sunday through Saturday. Example: Monday
- MONTHDAY - omits everything but the day of the month. Range: 1–31. Example: 9
- YEARDAY - transforms the day to the day of the year and omits the month, year, and time. Range: 1–366. Example: 160
- WEEK - transforms the date to the first day of the week as defined by Start-of-Week and omits the time. (Start-of-Week is a PDB property that you can specify with the %CPPDBOPT macro. By default, Start-of-Week is Sunday.) Example: 08Jun2003
- MONTH - omits the day and the time. Example: Jun2003
- YEARMONTH - omits everything but the month of the year. Range: 1–12. Example: 6
- QTR - transforms the month to the first month of the quarter and omits the day and time. Example: Apr2003

Note: You can change the width of the bar that appears on the 3-dimensional plot generated by %CPSPEC. If you specify PERIOD= ASIS, or 15MIN, or HOUR, the bar will be thinner in order to accommodate the increased number of points per line that is typical with shorter periods. △

PROCOPT=*string*
where *string* is any text that is permitted on the PROC statement of the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro. This allows expert SAS/GRAPH users to take advantage of options that are not supported by SAS IT Resource Management macros or new options for which SAS IT Resource Management has not yet added support.

For %CPPLOT1, %CPPLOT2, and %CPSPEC, the corresponding procedure in SAS/GRAPH is PROC GPLOT.

For %CPCCHRT and CPCHART, the corresponding procedure in SAS/GRAPH is PROC GCHART.

For %CPG3D, the corresponding procedure in SAS/GRAPH is PROC G3D.

For %CPTABRPT, the corresponding procedure in the SAS System is PROC TABULATE.

For example, you might specify an annotate data set:

```
%CPPLOT1(....,
         PROCOPT=ANNO=WORK.MYANNO,
         ....);
```

The default value is blank (no value).

For more details about what options are valid on the PROC GPLOT, PROC G3D, and PROC GCHART statements, see your SAS/GRAPH documentation. For information about PROC TABULATE, see your SAS System documentation.

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

SKIPMISS=YES | NO
specifies the type of lines to be used in the plot. If you specify SKIPMISS=YES (or SKIPMISS=Y), plot lines and areas *are* broken where there are missing values. If you specify SKIPMISS=NO (or SKIPMISS=N), plot lines and areas *are not* broken where there are missing values. The lines and areas are continuous.

The default value is YES.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

STAT=*statistic*
specifies what statistic to use in order to summarize data that spans the time interval that is specified by PERIOD=. The selected statistic is used for all variables in the report. *The default statistic is MEAN*. The STAT= parameter is not valid if PERIOD=ASIS. Valid values for STAT= are as follows:

CSS
is the sum of squares, corrected for the mean.

CV
is the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAX
is the maximum value for all observations.

MEAN
is the arithmetic average. Mean is one measure that is used to describe the center of a distribution of values.

MIN
is the minimum value for all observations.

N
(or COUNT) is the number of observations with nonmissing values for the variables that are being summarized.

NMISS
the number of observations with missing values for the variable being summarized.

RANGE
is the maximum of the difference between the maximum and minimum values for the variables, RANGE=MAX-MIN.

STD
is the standard deviation or square root of variance. Like the variance, it is a measure of the dispersion about the mean, but in the same unit of measure as the data.

STDERR
is the positive square root of the variance of a statistic.

SUM
is the sum of values for all observations.

USS
is the uncorrected sum of squares.

VAR
is a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: If you specify the BY= parameter, the statistics are calculated separately for each value of the BY variable or for each unique combination of the BY variable values, if you specify multiple BY variables. △

STMTOPT=*string*
where *string* is any text that is permitted as an option for the statement in the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro.

For %CPPLOT1 and %CPPLOT2, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPCCHRT and %CPCHART, the statement corresponds to the value of the TYPE parameter. For example, if TYPE=HBAR, then the statement will be the HBAR statement in PROC GCHART, and *string* will be added after the '/' in that statement.

For %CPSPEC, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPG3D, the corresponding statement in SAS/GRAPH is the SCATTER statement in PROC G3D.

For %CPTABRPT, the corresponding statement in SAS is the TABLE statement in PROC TABULATE.

If you want to customize the labeling and details of the axis for the group variable, you can specify an axis statement such as AXIS3 and then instruct %CPCHART to use it:

```
AXIS3 LABEL=(COLOR=BLUE);
%CPCHART(table,
         ....
         GROUP=gvar,
         STMTOPT=GAXIS=AXIS3,
         ....);
```

The default value is blank (no value).

For more details about what options are available for these statements, see the information about PROC GCHART, PROC G3D, and PROC GPLOT in your SAS/GRAPH documentation and PROC TABULATE in your SAS System documentation.

TABTYPE=INTERVAL | EVENT
specifies whether each observation in the data that is read into the table represents a specified interval of time or whether each observation was logged in response to an event, such as when a job began or ended. *If the data is in a view or data set in the detail, day, week, month, or year level of the active PDB, then the default is the value that is specified in the table's definition in the active PDB's data dictionary.*

INTERVAL
each observation represents an interval of time

EVENT
each observation represents an event.

TYPE=*plot-type*
specifies the type of plot to produce. *The default value is LINE.*

Valid values are described below. The value that you specify for the TYPE= parameter is passed to the SYMBOL statement. For more detailed information about how the SYMBOL statement uses these values, refer to the documentation of the SYMBOL Statement.

Note: The SYMBOL statement is a global option that can affect the performance of the SAS GPLOT procedure (PROC GPLOT) and SAS/GRAPH software. The SYMBOL statement is documented with the SAS/GRAPH software. For more information about the SYMBOL statement, see the documentation for your current release of the SAS system. △

Note: If the PERIOD= parameter is not set to ASIS and the TYPE= parameter is set to RC*xxxxx*, RL*xxxxx*, RQ*xxxxx*, or STD*xx*, then regression is performed on summarized data. Performing regression on summarized data does not conform to strict statistical standards. △

BOX
produces a box-and-whiskers plot that shows the 25th, 50th, and 75th percentiles of the values that are plotted. For more information, refer to the description of the INTERPOL=BOX option of the SYMBOL statement.

HILO*xx*
plots maximum and minimum values of combined data values as a vertical line. Refer to the description of the INTERPOL=HILO option of the SYMBOL statement.

JOIN
joins data points with a straight line. See also RC*xxxxx*, RL*xxxxx*, RQ*xxxxx*, SM*xxx*, and SPLINE*x* for joining techniques that use smoothing. Refer to the description of the INTERPOL=JOIN option of the SYMBOL statement.

LINE
joins adjacent data points with a straight line.

LINEDOT
joins adjacent data points with a straight line and uses a large dot to represent each data point. Refer to the description of the VALUE=DOT option of the SYMBOL statement.

LINESTAR
joins adjacent data points with a straight line and uses a large asterisk to represent each data point. Refer to the description of the VALUE=STAR option of the SYMBOL statement.

LINEX
joins adjacent data points with a straight line and uses a large X to represent each data point. Refer to the description of the VALUE=X option of the SYMBOL statement.

NEEDLE
plots the data points as a vertical line that drops from the point to the Y axis. Refer to the description of the INTERPOL=NEEDLE option of the SYMBOL statement.

PERCENT
plots each value of the variable as a percent of the total value for that variable. For example, the values 9 and 11 are converted to 45% and 55% and plotted at 45 and 55. This option is available only with SAS IT Resource Management software.

RC*xxxxx*
joins data points by using cubic regression as a smoothing technique. Refer to the description of the INTERPOL=RC option of the SYMBOL statement.

RL*xxxxx*
joins data points by using linear regression as a smoothing technique. Refer to the description of the INTERPOL=RL option of the SYMBOL statement.

RQ*xxxxx*
joins data points by using quadratic regression as a smoothing technique. Refer to the description of the INTERPOL=RQ option of the SYMBOL statement.

SCATTER
plots only the data points and does not join the points. Refer to the description of the INTERPOL=NONE option of the SYMBOL statement.

SM*xxx*
joins data points by using a cubic spline that minimizes a linear combination of the sum of squares of the residuals of fit and the integral of the square of the second derivative. The greater the *xxx* value is, the smoother the fitted curve will be. The value of *xxx* can range from 0P to 99P, or you can omit *xxx*. Refer to the description of the INTERPOL=SM option of the SYMBOL statement.

SPLINE*x*

joins data points by using cubic spline interpolation with continuous second derivatives. This method uses a piecewise third-degree polynomial for each set of two adjacent points. Refer to the description of the INTERPOL=SPLINE option of the SYMBOL statement.

STACK
plots the value or statistic in the first class and fills under the line, plots the value or statistic in the second class starting from the line for the first class and fills between the two lines, and so on. For example, the values 9 and 11 are plotted at 9 and 20 (the sum of 9 and 11).

STACKPCT
functions in the same way as STACK except that it plots each value or statistic of the variable as a percent of the total value or statistic for that variable. For example, the values 9 and 11 are converted to 45% and 55% and plotted at 45 and 100 (the sum of 45 and 55).

STD*xx*
uses a solid line to connect, for each X value, the mean Y value's statistic with the Y value's statistics that are two standard deviations away from the mean Y value's statistic. Refer to the description of the INTERPOL=STD option of the SYMBOL statement.

STEP*cxx*
plots the data with a step function. Refer to the description of the INTERPOL=STEP option of the SYMBOL statement.

UNIFORM=YES | NO
specifies the type of spacing to be used on the output report. *This parameter is called by several different macros, as follows*:

From %CPPRINT, the parameter specifies whether pages should use uniform spacing. If you specify UNIFORM=NO, then the pages do not have to be spaced uniformly and the layout of each page is optimized for the variable values that are printed on that page.

The default is uniform spacing (UNIFORM=YES).

From %CPPLOT1 and %CPPLOT2, the parameter specifies whether to use the same axis scale for all graphs when multiple graphs are produced.

UNIFORM=YES
specifies that multiple plots use the same axis scale for all the graphs.

UNIFORM=NO
specifies that multiple plots do not automatically use the same axis scale for all graphs. The axis is scaled appropriately for each individual graph. For more information, see the GPLOT procedure (PROC GPLOT) in the SAS/GRAPH documentation for your current release of SAS.

Note: If you specify the VAXIS= parameter on %CPPLOT1 or %CPPLOT2, then the value of the VAXIS= parameter overrides the value of the UNIFORM= parameter. △

VAXIS= AXIS*n* | *value-list* | LOW TO HIGH
specifies the value for the major tick marks on the left Y axis (the vertical axis), or an axis number that has been previously defined in a palette, or LOW TO HIGH. You can specify the value of the VAXIS= parameter in one of the following ways:

- the axis number, as in *axisn*, where *n* represents the axis number that you have already defined in the palette that you are currently using for this report
- a list of values for the axis
- or the words LOW TO HIGH (this is not case sensitive).

Note: Use LOW TO HIGH only if the analysis variables have non-negative values. △

Note: For %CPPLOT1, the setting VAXIS=LOW TO HIGH is meaningful. For %CPPLOT2, the setting VAXIS=LOW TO HIGH is not meaningful and is ignored. △

If you specify an axis number, then also specify the PALETTE= parameter in order to indicate the name of the palette that contains this axis.

These examples illustrate ways that you can specify the list of values (*value-list*):

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

Note: The values for VAXIS= must be specified as actual values, not formatted values. Thus, the tick marks for time values must be either a count of seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

Unless you specify LOW TO HIGH, the values for the VAXIS= parameter should reflect the range of values from your data that you want to represent in your report. Data that does not fall within the range of values set in the VAXIS= parameter will not be represented in the report. You will receive a warning message if all of your data does not fall within the range of values in the VAXIS= value list, but the report will continue processing.

For more information about the *value-list*, see the ORDER= parameter in the AXIS Statement section of the SAS/GRAPH documentation for your current version of SAS.

If you specify *LOW TO HIGH*, then multiple variables on the same plot will have their values mapped into a common range before plotting. The tick marks on the left vertical axis are labeled "LOW" to "HIGH" in this case, and the legend below the graph shows the actual axis range for each variable. The range of each variable is displayed by using the entire vertical space available on the graph. Thus, specifying VAXIS= LOW TO HIGH allows you to plot variables with widely differing values on a single graph without losing precision in the presentation for the smaller values. For example, you can plot CPU utilization (which varies between 0 and 1), disk I/O rate (which can range up into the thousands), and CPU queue length (which typically is less than ten) on one plot and still be able to see changes in all values. The VREF= and VZERO= parameters are normally specified with VAXIS=LOW TO HIGH. Because all values are mapped into the common range of 0-100, specifying VREF=0 25 75 100 can be used to provide reference lines. Specifying VZERO=Yes is also useful when specifying VAXIS=LOW TO HIGH, because this forces the lower limit of each variable to zero. This form is usually considered to provide a truer representation of the data than suppressing the zero on the graph. If you are plotting only one variable or two variables on a two-Y-axis plot, then VAXIS=LOW TO HIGH is ignored.

VREF=*value-list*

indicates values on the response axis where you want to draw one or more lines to use as reference lines. Separate the items in the *value-list* with one or more blanks.

value-list can also be in the form of:

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

The separator can be either a comma or blanks.

Note: This parameter is for horizontal bar charts, vertical bar charts, and block charts only. This parameter is ignored by pie charts and star charts. △

VZERO=NO | YES
specifies whether or not the tick marks on the left vertical axis should begin with zero. The VZERO request is ignored if negative values are present for the vertical variable or if the left vertical axis has been specified with the VAXIS= option. If you specify VZERO=NO or omit the VZERO= parameter, then the tick marks begin with a value that is appropriate to the range of the data values.

The default value is NO.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WEIGHT=*weight-variable*
specifies the alternate variable by which to weight (multiply) statistical calculations. If no WEIGHT= variable is specified and the table type is INTERVAL, then the variable DURATION is used as the weight.

For example, if most observations have a value of 15 for the weight variable and one observation has a value of 30 for the weight variable, then that observation has twice the weight of each of the other observations in any calculations.

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

XLAB=*label (obsolete; see LABELS=)*

specifies the label for the X axis of a plot. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the XVAR= variable's label in the PDB's data dictionary is used.

XVAR=*variable*

specifies the name of the variable for the X axis of a plot. The variable must be numeric. If you do not provide a variable's name, then the variable's name defaults to DATETIME.

%CPPLOT1 Notes

You can use this macro to produce two styles of reports:

- %CPPLOT1 Style 1
  - The analysis variable corresponds to the Y axis.
  - A separate line (or area, in a stack plot) is generated for each value of the class variable.
  - The variable that you specify for the X variable corresponds to the X axis.
  - You must specify one analysis variable and one class variable. You can specify one X variable. DATETIME is the default X variable.
- %CPPLOT1 Style 2
  - The analysis variable(s) corresponds to the Y axis. A separate line (or area, in a stack plot) is generated for each analysis variable.
  - The variable that you specify for the X variable corresponds to the X axis.
  - You must specify at least one analysis variable, but you can specify as many as 15 analysis variables. You can specify one X variable. DATETIME is the default X variable.
- For both report styles, you can specify *n* optional BY variables. No BY variables are required. If there is one BY variable, then a separate graph is generated for each value of the BY variable. If there is more than one BY variable, then a separate graph is generated for each unique combination of values of the BY variables.
- For both report styles, you can specify one optional statistic. The statistic is used to collapse (summarize and replace) data if the Summary Time Period is not AS IS. The default statistic is MEAN.
- For both report styles, you can specify one optional weight variable. In a table of type interval, if no weight variable is specified, then DURATION is used as the weight variable.

Overlay plots:

- If you want to plot multiple variables against the same X axis, then you can do this with %CPPLOT1.
- If you want to plot a single variable but want to have separate plot lines or plot symbols for different values of another variable (the x*y=z scenario in SAS/GRAPH), then specify the "z" variable as your class variable in %CPPLOT1.
- If you want two different scales for two different variables, then use %CPPLOT2.

Adjust the plot with VPOS and HPOS graphic options:

Under some circumstances, the plot produced by %CPPLOT1 can appear to be poorly formatted. This can be caused by different default values for SAS/GRAPH options VPOS and HPOS. The default values depend on the device driver being used. A lower HPOS can cause the labels to be written vertically, and this can consume most of the vertical space. To avoid this problem, increase HPOS or VPOS gradually until the picture is acceptable.

- To change the VPOS values from the SAS IT Resource Management GUI:

1. Type the following statement in the body of the Program Editor window:

```
goptions device=value
vpos=x;
```

   where VPOS=*x* specifies the number of rows in the graphics output area.
2. Submit the statement and rerun the report definition.

□ To change the HPOS values from the SAS IT Resource Management GUI:

1. Type the following statement in the body of the Program Editor window:

```
goptions device=value
hpos=x;
```

   where HPOS=*x* specifies the number of columns in the graphics output area.
2. Submit the statement and rerun the report definition.

□ To change the VPOS values from SAS IT Resource Management on z/OS in batch mode:

1. Add a GOPTIONS statement before the report definition in the batch job:

```
DEVICE=value VPOS=x
```

   where VPOS=*x* specifies the number of rows in the graphics output area.
2. Rerun the report definition.

□ To change the HPOS values from SAS IT Resource Management on z/OS in batch mode:

1. Add a GOPTIONS statement before the report definition in the batch job:

```
DEVICE=value HPOS=x
```

   where HPOS=*x* specifies the number of columns in the graphics output area.
2. Rerun the report definition.

The X axis is drawn to correspond to the data values being plotted, but you can use the %CPSETHAX macro to extend the X axis to provide a basic forecasting technique. For more information about %CPSETHAX, see "%CPSETHAX" on page 450.

%CPPLOT2

Generates a plot with two Y axes

%CPPLOT2 Overview

The %CPPLOT2 macro plots two Y variables on two Y axes, against one variable on the X axis.

The first Y variable specified is plotted against the left vertical axis, and the second Y variable specified is plotted against the right vertical axis. The two Y variables do not need to share the same unit of measurement or to have a similar range. The X axis can be any numeric variable and typically is DATETIME.

For more information on the underlying procedure, refer to the GPLOT Procedure in the SAS/GRAPH documentation for your current release of SAS.

%CPPLOT2 Syntax

%CPPLOT2(

dataset
,*analysis-variable-label-pairs*
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***BY-variable-list*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**HAXIS=***value-list* | AXIS*n* | contents of SAS AXISn statement>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LAXCOLOR=***color*>
<,**LAXLTYPE=***linetype*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**PERIOD=***time-interval*>
<,**PROCOPT=***string*>
<,**RAXCOLOR=***color*>
<,**RAXLTYPE=***linetype*>
<,**REDLVL=***PDB-level*>
<,**SKIPMISS=**YES | NO>
<,**SMALLDEV=***device-driver*>
<,**STAT=***statistic*>
<,**STMTOPT=***string*>
<,**TABTYPE=**INTERVAL | EVENT>
<,**UNIFORM=**YES | NO>
<,**VAXIS=***value-list* | AXIS*n* | Low to High>
<,**VAXIS2=***value-list* |AXIS*n*>
<,**VREF=***value-list*>
<,**VREF2=***value-list*>
<,**VZERO=**NO | YES>
<,**VZERO2=**NO | YES>
<,**WEBSTYLE=***style*>
<,**WEIGHT=***weight-variable*>
<,**WHERE=***where-expression*>
<,**XVAR=***variable*>);

Details

dataset
specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.

- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

analysis-variable-label-pairs
specifies a list of pairs. The first pair applies to the left vertical axis and the second pair applies to the right vertical axis. Each pair consists of the name of one analysis variable and the label for that analysis variable. Each label has a maximum length of 16 characters. The order for specifying these variables and labels is y1_var, y1_lab, y2_var, y2_lab.

You are required to specify two Y variables in this positional parameter. However, you are not required to specify labels when you use this parameter. In fact, using the LABELS= parameter to specify the labels is preferred.

The variables do not need to have the same unit of measure or similar ranges of values.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of a label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in a label.

Because this is a positional parameter, you must use a comma as a placeholder for the label when you specify more than one variable, but do not specify labels. For example, to specify two variables and no labels you could specify **`y_var1,,y_var2,KEYWORD=...`**. In this example, you have specified the left Y variable but no label for the left Y variable, and the right Y variable but no label for the right Y variable. You then specify any any keyword parameters, such as the LABELS= parameter, that you want to use in the report definition.

You can specify the labels by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the LABELS= parameter, then any labels that are specified by using this parameter are ignored. If labels are not specified by using this parameter and not specified by using the LABELS= parameter, then the variables' labels in the PDB's data dictionary are used.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **`dd:mmm:yyyy:hh:mm:ss`**, where **`dd`** is day, **`mmm`** is month, **`yyyy`** is year (in two-digit or four-digit notation), **`hh`** is hour (using a 24-hour clock), **`mm`** is minute, and **`ss`** is second.

- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*BY-variable-list*
lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*
lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

HAXIS=AXIS*n* | *value-list* | contents of SAS AXISn statement
specifies the value for the major tick marks on the X axis (the horizontal axis), or an axis number that has been previously defined in a palette, or the contents of a SAS AXIS statement without the word AXIS keyword or the ending semicolon. You can specify the value of the HAXIS= parameter in one of the following ways:

- the axis number, as in *axisn*, where *n* represents the axis number that you have already defined in the palette that you are currently using for this report
- a list of values for the axis

- the contents of a SAS AXISn statement (without the word AXIS at the beginning or the punctuation mark ";" at the end). In other words, anything that you can specify in an AXISn statement you can specify here.

If you specify an axis number, then also specify the PALETTE= parameter in order to indicate the name of the palette that contains this axis.

These examples illustrate ways that you can specify the list of values (*value-list*):

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

Note: The values for HAXIS= must be specified as actual values, not formatted values. Thus, the tick marks for time values must be either a count of seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

The values for the HAXIS= parameter should reflect the range of values from your data that you want to represent in your report. Data that does not fall within the range of values set in the HAXIS= parameter will not be represented in the report. You will receive a warning message if all of your data does not fall within the range of values in the HAXIS= value list, but the report will continue processing.

For more information about the *value-list*, see the ORDER= parameter in the AXIS Statement section of the SAS/GRAPH documentation for your current version of SAS.

Similarly, for more information about the contents of the AXIS statement, see the AXIS Statement section of the SAS/GRAPH documentation for your current version of SAS.

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*

specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **`welcome.htm`**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**`welcome.htm`** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **`HTMLURL=http://www.reporter.com/reports`** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*

specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named

www.reporter.com, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)

specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for

the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).
- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.
The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.
To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

  Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **`C:\`**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **`View Source`**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LAXCOLOR=*color*

indicates the color to use for the variable that is plotted on the left axis. The color must be a valid SAS color for the device that is used. For a list of available colors, see the SAS Companion for your operating environment.

LAXLTYPE= *linetype*
specifies the type of line to use for the variable that is plotted on the left axis. Valid line types are 0-48. If you specify 0, then the axis line is not drawn. The default is 1.

For more information on axis options, see the SYMBOL statement in the SAS/GRAPH software documentation for your current release of the SAS System.

LTCUTOFF=*time-of-cutoff*
specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:
If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

OUTDESC=*output-description*
if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*
for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT:* WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename | entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT*: G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see “Examples of the OUT*= Parameters” on page 560.

PALETTE=*palette-name*
specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **`sasuser.palette.win`** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section “Specifying/Editing/Viewing the Default Palette Definition” in the chapter “Reporting: Working with Palette Definitions” in the *SAS IT Resource Management User’s Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PERIOD=*time-interval*
is the summarization period for the report. *The default is ASIS*. For values other than ASIS, the values for the time period within a class or category are combined (as specified in the STAT= parameter) to form a single point on a plot, a bar on a chart, or a row or column in a table.

The following list provides the period name, the effect that it has on the report, and an example of how to specify the period.

- ASIS - leaves datetime in its present form. Example: 09Jun2003:14:37:25
- 15MIN - transforms the time to the first minute of the 15-minute period in the hour and retains the date. Example: 09Jun2003:14:30
- HOUR - transforms the time to the first minute of the hour and retains the date. Example: 09Jun2003:14:00
- 24HOUR - transforms the time to its hour component and omits the date. Range: 0–23. Example: 14
- DATE - omits the time and retains the date. Example: 09Jun2003
- WHOLEDAY - obsolete; use DATE.
- WEEKDAY - transforms the day to the day of the week (where 1=Sunday, 2=Monday, and so on) and omits the month, year, and time. Range: 1–7, which displays as Sunday through Saturday. Example: Monday
- MONTHDAY - omits everything but the day of the month. Range: 1–31. Example: 9
- YEARDAY - transforms the day to the day of the year and omits the month, year, and time. Range: 1–366. Example: 160
- WEEK - transforms the date to the first day of the week as defined by Start-of-Week and omits the time. (Start-of-Week is a PDB property that you can specify with the %CPPDBOPT macro. By default, Start-of-Week is Sunday.) Example: 08Jun2003
- MONTH - omits the day and the time. Example: Jun2003
- YEARMONTH - omits everything but the month of the year. Range: 1–12. Example: 6
- QTR - transforms the month to the first month of the quarter and omits the day and time. Example: Apr2003

Note: You can change the width of the bar that appears on the 3-dimensional plot generated by %CPSPEC. If you specify PERIOD= ASIS, or 15MIN, or HOUR, the bar will be thinner in order to accommodate the increased number of points per line that is typical with shorter periods. △

PROCOPT=*string*

where *string* is any text that is permitted on the PROC statement of the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro. This allows expert SAS/GRAPH users to take advantage of options that are not supported by SAS IT Resource Management macros or new options for which SAS IT Resource Management has not yet added support.

For %CPPLOT1, %CPPLOT2, and %CPSPEC, the corresponding procedure in SAS/GRAPH is PROC GPLOT.

For %CPCCHRT and CPCHART, the corresponding procedure in SAS/GRAPH is PROC GCHART.

For %CPG3D, the corresponding procedure in SAS/GRAPH is PROC G3D.

For %CPTABRPT, the corresponding procedure in the SAS System is PROC TABULATE.

For example, you might specify an annotate data set:

```
%CPPLOT1(....,
         PROCOPT=ANNO=WORK.MYANNO,
         ....);
```

The default value is blank (no value).

For more details about what options are valid on the PROC GPLOT, PROC G3D, and PROC GCHART statements, see your SAS/GRAPH documentation. For information about PROC TABULATE, see your SAS System documentation.

RAXCOLOR=*color*
indicates the color to use for the variable that is plotted on the right axis. The color must be a valid SAS color for the device that is used. For a list of available colors, see the SAS Companion for your operating environment.

RAXLTYPE=*linetype*
specifies the type of line to use for the variable that is plotted on the right axis. Valid line types are 0-48. If you specify 0, then the axis line is not drawn. The default is 1.

For more information on axis options, see the SYMBOL statement in the SAS/GRAPH software documentation for your release of the SAS system.

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

SKIPMISS=YES | NO
specifies the type of lines to be used in the plot. If you specify SKIPMISS=YES (or SKIPMISS=Y), plot lines and areas *are* broken where there are missing values. If you specify SKIPMISS=NO (or SKIPMISS=N), plot lines and areas *are not* broken where there are missing values. The lines and areas are continuous.

The default value is YES.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC*.

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

STAT=*statistic*
specifies what statistic to use in order to summarize data that spans the time interval that is specified by PERIOD=. The selected statistic is used for all variables in the report. *The default statistic is MEAN*. The STAT= parameter is not valid if PERIOD=ASIS. Valid values for STAT= are as follows:

CSS
is the sum of squares, corrected for the mean.

CV
is the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAX
is the maximum value for all observations.

MEAN
is the arithmetic average. Mean is one measure that is used to describe the center of a distribution of values.

MIN
is the minimum value for all observations.

N
(or COUNT) is the number of observations with nonmissing values for the variables that are being summarized.

NMISS
the number of observations with missing values for the variable being summarized.

RANGE
is the maximum of the difference between the maximum and minimum values for the variables, RANGE=MAX-MIN.

STD
is the standard deviation or square root of variance. Like the variance, it is a measure of the dispersion about the mean, but in the same unit of measure as the data.

STDERR
is the positive square root of the variance of a statistic.

SUM
is the sum of values for all observations.

USS
is the uncorrected sum of squares.

VAR
is a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: If you specify the BY= parameter, the statistics are calculated separately for each value of the BY variable or for each unique combination of the BY variable values, if you specify multiple BY variables. △

STMTOPT=*string*
where *string* is any text that is permitted as an option for the statement in the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro.

For %CPPLOT1 and %CPPLOT2, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPCCHRT and %CPCHART, the statement corresponds to the value of the TYPE parameter. For example, if TYPE=HBAR, then the statement will be the HBAR statement in PROC GCHART, and *string* will be added after the '/' in that statement.

For %CPSPEC, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPG3D, the corresponding statement in SAS/GRAPH is the SCATTER statement in PROC G3D.

For %CPTABRPT, the corresponding statement in SAS is the TABLE statement in PROC TABULATE.

If you want to customize the labeling and details of the axis for the group variable, you can specify an axis statement such as AXIS3 and then instruct %CPCHART to use it:

```
AXIS3 LABEL=(COLOR=BLUE);
%CPCHART(table,
         ....
         GROUP=gvar,
         STMTOPT=GAXIS=AXIS3,
         ....);
```

The default value is blank (no value).

For more details about what options are available for these statements, see the information about PROC GCHART, PROC G3D, and PROC GPLOT in your SAS/GRAPH documentation and PROC TABULATE in your SAS System documentation.

TABTYPE=INTERVAL | EVENT
specifies whether each observation in the data that is read into the table represents a specified interval of time or whether each observation was logged in response to an event, such as when a job began or ended. *If the data is in a view or data set in the detail, day, week, month, or year level of the active PDB, then the default is the value that is specified in the table's definition in the active PDB's data dictionary.*

INTERVAL
each observation represents an interval of time

EVENT
each observation represents an event.

UNIFORM=YES | NO
specifies the type of spacing to be used on the output report. *This parameter is called by several different macros, as follows*:

From %CPPRINT, the parameter specifies whether pages should use uniform spacing. If you specify UNIFORM=NO, then the pages do not have to be spaced uniformly and the layout of each page is optimized for the variable values that are printed on that page.

The default is uniform spacing (UNIFORM=YES).

From %CPPLOT1 and %CPPLOT2, the parameter specifies whether to use the same axis scale for all graphs when multiple graphs are produced.

UNIFORM=YES
specifies that multiple plots use the same axis scale for all the graphs.

UNIFORM=NO
specifies that multiple plots do not automatically use the same axis scale for all graphs. The axis is scaled appropriately for each individual graph. For more information, see the GPLOT procedure (PROC GPLOT) in the SAS/GRAPH documentation for your current release of SAS.

Note: If you specify the VAXIS= parameter on %CPPLOT1 or %CPPLOT2, then the value of the VAXIS= parameter overrides the value of the UNIFORM= parameter. △

VAXIS= AXIS*n* | *value-list* | LOW TO HIGH

specifies the value for the major tick marks on the left Y axis (the vertical axis), or an axis number that has been previously defined in a palette, or LOW TO HIGH. You can specify the value of the VAXIS= parameter in one of the following ways:

- the axis number, as in *axisn*, where *n* represents the axis number that you have already defined in the palette that you are currently using for this report
- a list of values for the axis
- or the words LOW TO HIGH (this is not case sensitive).

Note: Use LOW TO HIGH only if the analysis variables have non-negative values. △

Note: For %CPPLOT1, the setting VAXIS=LOW TO HIGH is meaningful. For %CPPLOT2, the setting VAXIS=LOW TO HIGH is not meaningful and is ignored. △

If you specify an axis number, then also specify the PALETTE= parameter in order to indicate the name of the palette that contains this axis.

These examples illustrate ways that you can specify the list of values (*value-list*):

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

Note: The values for VAXIS= must be specified as actual values, not formatted values. Thus, the tick marks for time values must be either a count of seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

Unless you specify LOW TO HIGH, the values for the VAXIS= parameter should reflect the range of values from your data that you want to represent in your report. Data that does not fall within the range of values set in the VAXIS= parameter will not be represented in the report. You will receive a warning message if all of your data does not fall within the range of values in the VAXIS= value list, but the report will continue processing.

For more information about the *value-list*, see the ORDER= parameter in the AXIS Statement section of the SAS/GRAPH documentation for your current version of SAS.

If you specify *LOW TO HIGH*, then multiple variables on the same plot will have their values mapped into a common range before plotting. The tick marks on the left vertical axis are labeled "LOW" to "HIGH" in this case, and the legend below the graph shows the actual axis range for each variable. The range of each variable is displayed by using the entire vertical space available on the graph. Thus, specifying VAXIS= LOW TO HIGH allows you to plot variables with widely differing values on a single graph without losing precision in the presentation for the smaller values. For example, you can plot CPU utilization (which varies between 0 and 1), disk I/O rate (which can range up into the thousands), and CPU queue length (which typically is less than ten) on one plot and still be able to see changes in all values. The VREF= and VZERO= parameters are normally specified with VAXIS=LOW TO HIGH. Because all values are mapped into the common range of 0-100, specifying VREF=0 25 75 100 can be used to provide reference lines. Specifying VZERO=Yes is also useful when specifying VAXIS=LOW TO HIGH, because this forces the lower limit of each variable to zero. This form is

usually considered to provide a truer representation of the data than suppressing the zero on the graph. If you are plotting only one variable or two variables on a two-Y-axis plot, then VAXIS=LOW TO HIGH is ignored.

VAXIS2=*value-list* | AXIS*n*
specifies the value for the major tick marks on the right Y axis (the right vertical axis) or an axis number that has been previously defined in a palette. You can specify the value in one of two ways:

- the axis number, as in *AXISn*, where *n* represents the axis number that you have already defined in the palette that you are currently using for this report
- a list of values for the axis.

If you specify an axis number, then you should also specify the PALETTE= parameter in order to indicate the name of the palette that contains this axis.

These examples illustrate ways that you can specify the *value-list*:

```
n n . . . n

or

n TO n <BY increment>

or

n n . . . n TO n
   <BY increment> <n . . . n>
```

Note: The values for VAXIS2= must be specified as actual values, not formatted values. Thus, the tick marks for time values must be either a count of seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

The values for the VAXIS2= parameter should reflect the range of values from your data that you want to represent in your report. Data that does not fall within the range of values set in the VAXIS2= parameter will not be represented in the report. You will receive a warning message if all of your data does not fall within the range of values in the VAXIS2= value list, but the report will continue processing.

For more information about the *value-list*, see the ORDER= parameter in the AXIS Statement section of the SAS/GRAPH documentation for your current release of SAS.

VREF=*value-list*
indicates values on the response axis where you want to draw one or more lines to use as reference lines. Separate the items in the *value-list* with one or more blanks.

value-list can also be in the form of:

```
n n . . . n

or

n TO n <BY increment>

or
n n . . . n TO n
    <BY increment> <n . . . n>
```

The separator can be either a comma or blanks.

Note: This parameter is for horizontal bar charts, vertical bar charts, and block charts only. This parameter is ignored by pie charts and star charts. △

VREF2=*value-list*
indicates values on the right Y axis (right vertical axis) where you want to draw one or more horizontal lines to use as reference lines. Separate items in the *value-list* with blanks.

VZERO=NO | YES
specifies whether or not the tick marks on the left vertical axis should begin with zero. The VZERO request is ignored if negative values are present for the vertical variable or if the left vertical axis has been specified with the VAXIS= option. If you specify VZERO=NO or omit the VZERO= parameter, then the tick marks begin with a value that is appropriate to the range of the data values.

The default value is NO.

VZERO2=NO | YES
specifies whether or not the tick marks on the right vertical axis should begin with zero. The VZERO2= request is ignored if negative values are present for the vertical variable or if the right vertical axis has been specified with the VAXIS2= option. If you specify VZERO2=NO or omit the VZERO2= parameter, then the tick marks begin with a value that is appropriate to the range of the data values.

The default value is NO.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WEIGHT=*weight-variable*
specifies the alternate variable by which to weight (multiply) statistical calculations. If no WEIGHT= variable is specified and the table type is INTERVAL, then the variable DURATION is used as the weight.

For example, if most observations have a value of 15 for the weight variable and one observation has a value of 30 for the weight variable, then that observation has twice the weight of each of the other observations in any calculations.

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the

ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

XVAR=*variable*
specifies the name of the variable for the X axis of a plot. The variable must be numeric. If you do not provide a variable's name, then the variable's name defaults to DATETIME.

%CPPLOT2 Notes

For this macro and report style:

- One analysis variable corresponds to the left Y axis and the other analysis variable corresponds to the right Y axis. You must specify two analysis variables.
- The X variable must be numeric. Typically, the X variable is DATETIME.
- You can specify *n* optional BY variables. No BY variables are required. If there is one BY variable, then a separate graph is generated for each value of the BY variable. If there is more than one BY variable, then a separate graph is generated for each unique combination of values of the BY variables.
- You can specify one optional statistic. The statistic is used to collapse (summarize and replace) data if the Summary Time Period is not AS IS. The default statistic is MEAN.

- You can specify one optional weight variable. In a table of type interval, if no weight variable is specified, then DURATION is used as the weight variable.

Overlay plots:

- If you want to plot multiple variables against the same X axis, then you can do this with %CPPLOT1.
- If you want to plot a single variable but want to have separate plot lines or plot symbols for different values of another variable (the x*y=z scenario in SAS/GRAPH), then specify the "z" variable as your class variable in %CPPLOT1.
- If you want two different scales for two different variables, then use %CPPLOT2.

Adjust the plot with VPOS and HPOS graphic options:

Under some circumstances, the plot produced by %CPPLOT2 can appear to be poorly formatted. This may be caused by different default values for SAS/GRAPH options, VPOS and HPOS. The default values depend on the device driver being used. A lower HPOS may cause the labels to be written vertically, which may consume most of the vertical space. To circumvent this problem, increase HPOS or VPOS gradually until the picture is acceptable.

- To change the VPOS values from the SAS IT Resource Management GUI:
  1. Type the following statement in the body of the SAS Program Editor window:

     ```
     goptions device=value vpos=x;
     ```

     where VPOS=*x* specifies the number of rows in the graphics output area.
  2. Submit it and rerun the report definition.
- To change the HPOS values from the SAS IT Resource Management GUI:
  1. Type the following statement in the body of the SAS Program Editor window:

     ```
     goptions device=value hpos=x;
     ```

     where HPOS=*x* specifies the number of columns in the graphics output area.
  2. Submit it and rerun the report definition.

- To change the VPOS values from SAS IT Resource Management on z/OS in batch mode:
  1. Add a GOPTIONS statement before the report definition in the batch job:

     ```
     goptions DEVICE=value VPOS=x;
     ```

     where VPOS=*x* specifies the number of rows in the graphics output area.
  2. Rerun the report definition.
- To change the HPOS values from SAS IT Resource Management on z/OS in batch mode:
  1. Preceding the report definition in the batch job, add the following GOPTIONS statement:

     ```
     goptions DEVICE=value HPOS=x;
     ```

     where HPOS=*x* specifies the number of columns in the graphics output area.
  2. Rerun the report definition.

%CPPRINT

Prints the specified table in a simple rectangular listing

%CPPRINT Overview

The %CPPRINT macro enables you to print the contents of a table that is in a PDB. Each analysis variable generates a separate column of the report. You can specify up to 30 analysis variables.

%CPPRINT Syntax

%CPPRINT(
dataset
,*var1*<,...,*var30*>
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***BY-variable-list*>
<,**DOUBLE=**YES | NO>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**HEADING=**HORIZONTAL | VERTICAL>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**ID_NAM=***variable*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=** *URL-specification*>
<,**LABEL=**YES | NO>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**N=**YES | NO>
<,**NOOBS=**YES | NO>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PAGEBY=***variable*>
<,**PALETTE=***palette-name*>
<,**REDLVL=***PDB-level*>
<,**ROUND=**YES | NO>
<,**ROWS=**PAGE>
<,**SMALLDEV=***device-driver*>
<,**SPLIT=***'split-character'*>
<,**SUM=***variable-list*>
<,**SUMBY=***variable*>
<,**UNIFORM=**YES | NO>
<,**WEBSTYLE=***style*>
<,**WHERE=***where-expression*>
<,**WIDTH=**FULL | MINIMUM | UNIFORM | UNIFORMBY>);

Details

dataset
specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

var1<,...,var30>
specifies the variables whose values will be printed. You must specify at least one variable, but you can specify a maximum of 30 variables.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **dd:mmm:yyyy:hh:mm:ss**, where **dd** is day, **mmm** is month, **yyyy** is year (in two-digit or four-digit notation), **hh** is hour (using a 24-hour clock), **mm** is minute, and **ss** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.

- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*BY-variable-list*

lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

DOUBLE=YES | NO
specifies whether the printed output should be single-spaced or double-spaced. For double-spaced output, specify DOUBLE=YES. *The default is single-spaced output (DOUBLE=NO).*

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*
lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

HEADING=HORIZONTAL | VERTICAL
specifies how to print the column headings. Specify HEADING=HORIZONTAL or HEADING=VERTICAL in order to print horizontal or vertical headings for this printed output. *The default is to print the column headings according to the best fit for the page size.*

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

ID_NAM=*variable*
specifies the name of the ID variable. When you specify this parameter, this variable is used to identify observations in place of the default observation number.

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the

macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **welcome.htm** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the

value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABEL=YES | NO
specifies whether to use variable names or labels as column headings. If you specify LABEL=NO, then variable names (instead of variable labels) are used as column headings. *The default is LABEL=YES, which means that your variable labels are used as column headings.*

LABELS=(*var-label-pairs*)
specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*

specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).
- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web

browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

  Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **`C:\`**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*

specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

N=YES | NO
specifies whether or not to print, at the end of the output, the total number of observations and the number of observations in each BY group (if a BY variable list is used). Specify N=YES to print this information at the end of the output. *The default is N=NO, which does not print this information at the end of the output.*

NOOBS=YES | NO
specifies whether each printed observation should be identified by its observation number. If you specify NOOBS=YES, then the printed observations are not identified by their observation numbers. If you specify the ID_NAM= parameter, then NOOBS=YES is implied. *The default is NOOBS=NO, which identifies observations by the value of the variable that is specified in the ID_NAM= parameter if the ID_NAM= parameter is specified, or by the observation number if the ID_NAM= parameter is not specified.*

OUTDESC=*output-description*
if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*
for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS

catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT*: WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*

specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*

- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename* | *entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT:* This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.

- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

PAGEBY=*variable*
specifies when to start a new page (if the OUTMODE= parameter is not equal to WEB, or if OUTMODE=WEB and the value of HTMLDIR= is a PDS) or a new file (if OUTMODE=WEB and the value of HTMLDIR= is a directory or folder). A new page or file will start when the value of this variable (or a variable that precedes it in the BY variables list) changes. The PAGEBY variable must also be specified in the BY variables list.

If the PAGEBY= parameter is specified, the %CPPRINT macro checks how many pages or files will be generated. If the number is greater than a limit, then no pages or files are generated, the macro terminates and writes an ERROR message to the log, and the job continues. If you want to change the limit, add this statement before the call to the macro:

```
%let cp_pglim = n ;
```

where *n* is the new limit. By default, *n* is 500.

Suppose that OUTMODE=WEB, the PAGEBY= variable is not specified, and the filename is *rptname*.htm. Then, if OUTMODE=WEB and the PAGEBY= variable is specified and the report is directed to a folder or directory, the filenames take the form *rptname*_page*n*.htm. (For example, page 3 of a report named *myrpt* would be *myrpt_page3.htm*.) For more information about report output filenames, see the OUTNAME= parameter.

Note: On z/OS, if the following three conditions are all true, then you might want to have the HTMLDIR= parameter direct the reports to a directory in the UNIX File System area instead of directing the report to a PDS:

- you are calling the %CPPRINT macro
- the OUTMODE= parameter is set to WEB
- you are using the PAGEBY= parameter.

When the report is directed to a PDS, the report is a very large single file without the typical filtering. When the report is directed to a directory, the report becomes multiple small files with typical filtering. △

PALETTE=*palette-name*
specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **sasuser.palette.win** if the palette is stored in your SASUSER library, in a

palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

REDLVL=*PDB-level*

specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △
- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL=

parameter that was specified in the underlying report definition(s) being invoked.

ROUND=YES | NO
indicates whether or not values for numeric variables should be rounded. Specify ROUND=YES to round the printed values for numeric variables to two decimal places, if a format is not already associated with the variable. Variables are rounded before they are summed. *The default is ROUND=NO, which means that the variable values are not rounded.*

ROWS=PAGE
indicates that you want to print only one row of variables per page. *The default is to split the page into sections when all variables cannot be printed in one row.*

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

SPLIT='*split-character*'
specifies a character to use in order to split column headings. You must enclose the character in single quotation marks. If you specify this parameter, then the column headings split at the location of this character in the variable label.

SUM=*variable-list*
specifies the list of variables to sum.

SUMBY=*variable*
specifies when to print subtotals. Subtotals will print when the value of this variable (or a variable that precedes it in the BY variables list) changes. The SUMBY variable must also be specified in the BY variables list. If the SUMBY= parameter is specified but the SUM= parameter is not specified, then all numeric variables except BY and ID variables are summed.

UNIFORM=YES | NO
specifies the type of spacing to be used on the output report. *This parameter is called by several different macros, as follows*:

From %CPPRINT, the parameter specifies whether pages should use uniform spacing. If you specify UNIFORM=NO, then the pages do not have to be spaced uniformly and the layout of each page is optimized for the variable values that are printed on that page.

The default is uniform spacing (UNIFORM=YES).

From %CPPLOT1 and %CPPLOT2, the parameter specifies whether to use the same axis scale for all graphs when multiple graphs are produced.

UNIFORM=YES
specifies that multiple plots use the same axis scale for all the graphs.

UNIFORM=NO
specifies that multiple plots do not automatically use the same axis scale for all graphs. The axis is scaled appropriately for each individual graph. For

more information, see the GPLOT procedure (PROC GPLOT) in the SAS/GRAPH documentation for your current release of SAS.

Note: If you specify the VAXIS= parameter on %CPPLOT1 or %CPPLOT2, then the value of the VAXIS= parameter overrides the value of the UNIFORM= parameter. △

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE

expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

WIDTH=FULL | MINIMUM | UNIFORM | UNIFORMBY
specifies the method of formatting pages based on column widths. FULL uses the full width of the format that is assigned to that variable. MINIMUM uses the minimum possible column width. UNIFORM formats the columns uniformly across all pages. UNIFORMBY formats the columns uniformly within a BY group. *The default is to format each page according to its own best fit.*

%CPPRINT Notes

If you specify OUTMODE=WEB for this macro, you should also specify the OUTNAME= parameter so that the file created is outname.htm. If you do not specify a value for the OUTNAME= parameter, then the filename defaults to the existing report definition name (for a saved report definition) or to PRTRPT*n*.htm (for a new report definition).

Additionally, if you use OUTMODE=WEB with this report macro, then reports with the same OUTDESC= (description) cannot be displayed on the same Web page. Therefore, either avoid duplicate descriptions or specify a different HTML directory for reports that have the same description.

You can use this macro to produce a report in which you can do the following:

- You can specify 1–30 analysis variables. A separate column of the report is generated for each analysis variable.
- You can specify *n* optional BY variables. No BY variables are required. If there is one BY variable, then a separate section of the report is generated for each value of the BY variable. If there is more than one BY variable, then a separate section of the report is generated for each unique combination of values of the BY variables.
- You can specify one optional variable, by using the IDNAME= parameter. If an ID variable is specified, then each observation in the report is identified by the ID variable's value instead of the observation's number.

%CPRUNRPT

Runs a saved report definition or several report definitions in a folder

%CPRUNRPT Overview

The %CPRUNRPT macro enables you to specify the name of an existing report definition or report folder and to run the definition (or all definitions in a folder) in batch mode. You can temporarily override certain subsetting and output characteristics of the defined report(s) when using %CPRUNRPT.

Note: If the call to the %CPRUNRPT macro and the stored report definition both specify the same parameter, then the value that is specified in the call to the %CPRUNRPT takes precedence.

The parameters set in %CPRUNRPT are temporary; they are not saved and must be respecified if they are needed in later executions of the reports. △

%CPRUNRPT Syntax

%CPRUNRPT(
<*rptname*>
<,**BEGIN=***SAS-datetime-value*>
<,**END=***SAS-datetime-value*>
<,**FOLDER=***report-folder*>
<,**GBLWHERE=***where-clause*>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**REDLVL=***PDB-level*>
<,**RSUBMIT=**YES | NO>
<,**SMALLDEV=***device-driver*>
<,**WEBCLR=**YES | NO>
<,**WEBSTYLE=***style*>
<,**WHERE=***where-expression*>);

Details

rptname

specifies the name of one or more existing report definitions that you want to run. If you specify multiple names, then use one or more blanks to separate the names in the list. Do not specify *.sas* as part of the name.

This parameter is not required, but you must specify either this parameter or the FOLDER= parameter (or you can specify both parameters). If you specify this parameter and not the FOLDER= parameter, then all report folders are searched in order to find the report definition name that is specified in this parameter. The folders are searched in the order in which they appear in the Report Folders list in the Manage Report Definitions window, within the SAS IT Resource Management GUI for UNIX and Windows environments. The %CPRUNRPT macro runs the first report definition that is found with this name.

That list is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a particular folder with the FOLDER= parameter.

For additional concepts, refer to the "Using the Manage Report Definitions Tool" section in the "Report Concepts and Tools" chapter in the *SAS IT Resource Management User's Guide*.

If you do not specify this parameter and you specify a folder name, then this macro attempts to run all report definitions in the specified folder.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **dd:mmm:yyyy:hh:mm:ss**, where **dd** is day, **mmm** is month, **yyyy** is year (in two-digit or four-digit notation), **hh** is hour (using a 24-hour clock), **mm** is minute, and **ss** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FOLDER=*report-folder*
specifies the name of the report folder that contains the report definitions that you want to run.

Note: *report-folder* is the name of a SAS data set or view specified as *libref.name*, where *libref* must be defined before this macro is called and *name* is the name of the data set or view. For example, you might specify FOLDER=PGMLIB.ITSVRPT, where PGMLIB is a libref and ITSVRPT is the data set name. △

This parameter is not required, but you must specify either this parameter or the rptname parameter (or you may specify both parameters).

If you specify this parameter and not the *rptname* parameter, the macro attempts to run all report definitions in the folder.

the chapter "Reporting: Working with Report Definition Folders" in the *SAS IT Resource Management User's Guide*.

To view the list of report folders, select the Manage Report Definitions task on the Reporting tab in the SAS IT Resource Management GUI.

Note: %CPRUNRPT can be executed on z/OS. However, this macro runs only report folders or report definitions that have been transferred to z/OS from the SAS IT Resource Management client host (by uploading through SAS/CONNECT). △

GBLWHERE=*where-clause*
specifies a WHERE value (without the WHERE keyword), which temporarily overrides any currently active global WHERE clause and thus appends to (by

means of "SAME AND") each individual report definition's local WHERE, if any. For more information, see "Global and Local Macro Variables" on page 6.

The GBLWHERE= value can be blank or any legal SAS WHERE clause (without the WHERE keyword).

Note: Do not use the ampersand (&) to mean AND in WHERE expressions. △

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhdir=c:\www\hreports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the

z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the

files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on

"%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **welcome.htm** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML

files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LARGEDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).
- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does

provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

  Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*
specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the

specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

OUTDESC=*output-description*

if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*

- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*

for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT*: WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename* | *entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

PALETTE=*palette-name*

specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **sasuser.palette.win** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △
- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

RSUBMIT=YES | NO
specifies whether or not to run the report definition(s) on the local or remote host. If the active PDB is local (that is, if you did not specify the REMPROF= parameter in your most recent call to the %CPSTART macro), then do not specify this parameter. If the active PDB is remote, then you can specify this parameter. *This parameter is not required, and the default value is NO.*

- RSUBMIT=YES runs the report definition(s) on the remote PDB's host and displays the results or output on your local host.

  Do not wrap the %CPRUNRPT call with *rsubmit;* and *endrsubmit;* because the %CPRUNRPT call will wrap itself if RSUBMIT=YES.
- RSUBMIT=NO runs the report definition(s) on your local host and displays the results or output on your local host. In this case, the remote PDB is read, the remote data is downloaded, and the report definition(s) run locally and display the results or output on your local host.

  However, if the %CPRUNRPT call is wrapped with *rsubmit;* and *endrsubmit;*, the call will run on the remote PDB's host and display the results or output on your local host.

You can use the RSUBMIT= parameter only with the %CPRUNRPT macro.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

WEBCLR=YES | NO
specifies whether or not to clear reports from the SAS catalog that is specified with the OUTLOC= parameter and the Web output directories (or PDSs) that are specified with the HTMLDIR= parameter and, optionally, with the IMAGEDIR= parameter. (The **info.htm** file and a few other help and information files are not deleted.)

For the %CPRUNRPT macro, the default value of this parameter is NO. For the %CPXHTML macro, the default value of this parameter is YES.

Note:

- For the %CPRUNRPT macro, you can clear out all the reports by specifying WEBCLR= or by invoking the %CPWEBINI macro. Alternatively, you can clear out selected reports by invoking the %CPMANRPT macro.
- For the %CPXHTML macro, you can clear out all the reports by specifying WEBCLR= or by invoking the %CPWEBINI macro. However, you cannot clear out selected reports by invoking the %CPMANRPT macro.

△

If you specify WEBCLR=YES, then any existing reports in the specified catalog and directory are deleted before any new reports are written to the catalog and directory. However, the static files are not deleted when you specify WEBCLR=YES. For more information about the static files, see the topic "Customizing a Report Gallery's Static Files" in the chapter "Reporting: Working with Galleries" in the *SAS IT Resource Management User's Guide*.

Note: For the %CPRUNRPT and %CPXHTML macros, if you specify WEBCLR=YES and OUTMODE=WEB, but no OUTLOC= parameter is specified, then any reports in the OUTLOC= default location will be cleared. △

If you specify WEBCLR=NO, then existing reports in the specified catalog and directory are not deleted; new reports are added to the existing set of reports. You,

the user, assume responsibility for clearing reports. For example, you might specify YES once a week, or you might clear the reports only when a Web report macro fails because of an out-of-time condition, an out-of-space condition, or some other problem that is indicated by an error message in the SAS log.

Note: You can also use the %CPWEBINI macro as an alternate way to clear reports from the Web catalog and directory. In both %CPWEBINI and WEBCLR=, if you clear the catalog, then you can clear or not clear the directory; however, if you clear the directory, then you must clear the catalog. (If you want to delete some but not all reports, then use the %CPMANRPT macro instead of WEBCLR= or %CPWEBINI.) △

This parameter is valid only if you specify OUTMODE=WEB or the macro is %CPXHTML.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE.

(*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

%CPRUNRPT Notes

If you specify one or more report definitions explicitly, %CPRUNRPT runs only those. If you leave the list empty, %CPRUNRPT runs all the report definitions in that folder that refer to the views that are available in the active PDB.

In the folder, the definition directs the report to the GRAPH window. You can use parameters on the %CPRUNRPT macro to override the direction — that is, direct the report elsewhere. As a result, the very same report definition can be used interactively and in batch/background mode without change. If you do change the report definition, the change is expressed everywhere the report definition is used.

%CPSETHAX

Sets the horizontal axis for forecasting reports

%CPSETHAX Overview

The %CPSETHAX macro creates a global macro variable whose value is an expression that extends the axis for a regression plot, beyond the values of the input data. This macro variable can then be referenced and used in other SAS IT Resource Management reporting macros in order to extend the horizontal axis for the reports.

%CPSETHAX Syntax

%CPSETHAX(
DATA=*libref.name*
,**DSVAR=***var-name*
<,**EXTEND=***number*>
<,**MACVAR=***macro-variable*>
<,**NUMTICS=***number*>
<,**_RC=***macro-var-name*>
<,**WHERE=***expression*>

<,**ZEROMIN=**YES | NO>);

Details

DATA=*libref.name*
specifies the name of the input data set or data view that contains the variable that is specified by the DSVAR= parameter. The name must be specified in the form *libref.name*. The *libref* points to the library that contains the data set or view; it must be defined before this macro is called. The *name* is the data set name or data view name.

This parameter is required.

DSVAR=*var-name*
specifies the name of the variable that is displayed on the horizontal axis in your report, which is the axis that you want to extend. This variable must be available in the input data set or view that is specified by the DATA= parameter.

This parameter is required.

EXTEND=*number*
specifies the number to use to extend the horizontal axis. This number is used to calculate the distance beyond the current data values that the horizontal axis is to be extended. You can specify any number between 0 and 10. *This parameter is not required, and the default value is 1.5.*

The extension value is calculated by multiplying the value of the EXTEND= parameter by the current range of the data values in the input data set, and then adding that value to the current left axis value. For example, if you specify EXTEND=1.5 and your data values range from 100 to 500, then the right end of the horizontal axis would be extended (or recalculated) to a value of 700, which is beyond your current data range of 500. The extension value is calculated as shown below. In the example, the LeftEndPointOfAxis (or minimum data value) is 100, the RightEndPointOfAxis (or maximum data value) is 500, and the ExtendVal is 1.5. After the extension is applied, the NewRightEndPoint of the horizontal axis is 700.

```
100 + ((500-100) * 1.5) = 700
```

The values shown in this example represent the following:

```
LeftEndPointOfAxis +
((RightEndPointOfAxis - LeftEndPointOfAxis) * ExtendVal) =
NewRightEndPoint
```

Note: ZEROMIN= affects only the beginning point of the axis (minimum data value). It does not affect the calculations of the data range (max-min). The new right endpoint of the extended horizontal axis is calculated as

```
minimum data value = ((maximum data value - minimum data value) * EXTEND=
value)
```

The default value of ZEROMIN= is YES, which forces the minimum value of the data (the left endpoint of the axis) to be zero. In order to use a value other than zero as the left endpoint, you must specify ZEROMIN=NO. This sets the lowest value in the range of data as the left endpoint on the horizontal axis. If your variable is DATETIME, then the default value of the ZEROMIN= parameter automatically changes to ZEROMIN=NO, which prevents the left endpoint on the horizontal axis from defaulting to 01Jan1960:00:00. △

MACVAR=*macro-variable*
specifies the name of the macro variable that you will reference in your report, to extend the axis. *This parameter is not required. The default value is CPHAXIS.*

NUMTICS=*number*
specifies the number of tick marks to display on the horizontal axis. You can specify a number between 0 and 25.
This parameter is not required, and the default value is seven.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △
If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.
For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.
You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).
There is no default value for the name of the macro variable.

WHERE=*expression*
specifies the expression that will be used to subset the input data set. Specify this parameter only if you want to subset the input data. *This parameter is not required, and by default the input data is not subset.*

ZEROMIN=YES | NO
indicates whether to use zero as the minimum value on the horizontal axis. If you specify ZEROMIN=NO, then the axis begins at the actual minimum value in the input data set. *If you specify ZEROMIN=YES, which is the default, then zero is used as the starting point on the axis.* This parameter is not required.

Note: If your variable is DATETIME, then the default value of ZEROMIN= automatically defaults to NO to prevent the left endpoint on the horizontal axis from defaulting to 01Jan1960:00:00. △

%CPSETHAX Example

The following example creates a global macro variable that is set to the horizontal axis expression. The macro variable is then used to extend the axis for a regression plot, beyond the values of the input data.

```
* Produce a regression forecast of network packet rate based
* on transaction count.  Use %CPSETHAX to force the horizontal
* axis on the plot beyond the actual data values. %CPSETHAX
* sets macro global variable CPHAXIS, which is then used to
* set CPHAXIS (referenced by the VREF= parm on the %CPPLOT1
* report macro);
```

```
%CPSETHAX( data    = week.pcsglb,
           dsvar   = glbtrnm,
           macvar  = tranaxis,
           where   = shift='1');
title1 "Net Traffic Forecast from Transactions";
%CPPLOT1(
        pcsglb
       ,glbpkrt,
       ,where=shift='1'
       ,type=rqclm95
       ,xvar=glbtrnm
       ,by=machine
       ,redlvl=detail
       ,vref= 0 &tranaxis
       ,vzero=no
       ,labels=(machine="Machine"
                glbtrnm="Transactions/Min"
                glbpkrt="Packets/Second")
        );
```

%CPSPEC

Creates a three-dimensional plot with color as the third dimension (spectrum plot)

%CPSPEC Overview

The %CPSPEC macro generates a two-dimensional surface plot. In preparation for generating the plot, the values of the analysis variable are grouped by percentiles or by specified data values and then each group is represented on the graph by a color. The plot resembles a multiline spectrograph, with the variable DATETIME corresponding to the X axis.

If you do not specify a class variable, you can specify a maximum of 15 analysis variables and labels for this report. A separate row on the plot is generated for each analysis variable.

If you do specify a class variable (by using the CL_NAM= parameter), then you can specify only one analysis variable (an label). In this case, a separate row is generated for each value of the class variable.

By default, the colors on the graph represent the data values of the analysis variables by using a cool-to-hot spectrum, with blue indicating cool or low values, green to orange indicating warm or midrange values, and red indicating hot or high values. However, you can use the COLORLST= parameter to change the colors that are used in your spectrum plot.

%CPSPEC Syntax

%CPSPEC(
dataset
,*variable-label-pairs*
<,**AXIS=***value-list* | AXIS*n* | contents of SAS AXISn statement>
<,**BEGIN=***SAS-datetime-value*>
<,**BIGYAXIS=**YES | NO>

<,**BY=***BY-variable-list*>
<,**CL_LAB=***label*> (obsolete; use LABELS=)
<,**CL_NAM=***variable-name*>
<,**COLORLST=***value-list*>
<,**CTEXT=***color*>
<,**DATAVALS=***value-list*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LGNDFMT=***format-name* | DATA>
<,**LTCUTOFF=***time-of-cutoff*>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**PERCENTS=***value-list*>
<,**PERIOD=***time-interval*>
<,**PROCOPT=***string*>
<,**REDLVL=***PDB-level*>
<,**SMALLDEV=***device-driver*>
<,**STAT=***statistic*>
<,**STMTOPT=***string*>
<,**TABTYPE=**INTERVAL | EVENT>
<,**WEBSTYLE=***style*>
<,**WEIGHT=***weight-variable*>
<,**WHERE=***where-expression*>
<,**YVAL=***number*>
<,**XVAR=***variable*>);

Details

dataset

specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a

libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

variable-label-pairs
specifies a list of pairs. (For %CPCHART, you can use only one pair.) Each pair consists of the name of one analysis variable and the label for that analysis variable. Each label has a maximum length of 16 characters.

The maximum number of pairs that you can specify is 15, depending on the type of report that you are running. *For a detailed explanation of the variables that are displayed on each axis and the maximum number of variables that can be specified for the report, refer to the "Overview" section of the macro description.*

If you specify multiple analysis variables, then the unit of measure for all of the analysis variables must be the same and the range of values should be similar.

The analysis variable is typically displayed on the left (Y) axis. However, certain combinations of parameters might affect the axis on which the analysis variable is displayed or the number of variables that can be specified, depending on the report macro that you use.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of a label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in a label.

You are required to specify at least one variable in this positional parameter. However, you are not required to specify labels when you use this parameter. In fact, using the LABELS= parameter to specify the labels is preferred. Because this is a positional parameter, you must use a comma as a placeholder for the label when you specify more than one variable but do not specify labels. For example, to specify three variables and no labels, you could specify **`var1,,var2,,var3,KEYWORD=...`**. In this example, you have specified variable 1 but no label for variable 1, variable 2 but no label for variable 2, and variable 3 but no label for variable 3. You then specify any keyword parameters, such as the LABELS= parameter, that you want to use in the report definition.

You can specify the labels by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the LABELS= parameter, then any labels that are specified by using this parameter are ignored. If labels are not specified by using this parameter and not specified by using the LABELS= parameter, then the variables' labels in the data dictionary are used.

AXIS=AXIS*n* | *value-list* | contents of SAS AXISn statement
specifies the value for the major tick marks on the X axis (the horizontal axis), or an axis number that has been previously defined in a palette, or the contents of a SAS AXIS statement without the word AXIS or the ending semicolon. You can specify the value of the AXIS= parameter by specifying one of the following:

- the axis number, as in *axisn*, where *n* represents the axis number that you already defined in the palette that you are currently using for this report
- a list of values for the axis
- the contents of a SAS AXISn statement (without the word AXIS at the beginning or the semicolon at the end). In other words, anything that you can specify in an AXISn statement you can specify here.

If you specify an axis number, then also specify the PALETTE= parameter in order to indicate the name of the palette that contains this axis.

These examples illustrate ways that you can specify the list of values (*value-list*):

```
n n . . . n
```

or

```
n TO n <BY increment>
```

or

```
n n . . . n TO n
    <BY increment> <n . . . n>
```

Note: The values for AXIS= must be specified as actual values, not formatted values. Thus, the tick marks for time values must be either a count of seconds or a SAS time constant, such as '12:20:01.8't. You must specify percentages as decimal values, such as .10 to .80 for 10% to 80%. △

The values for the AXIS= parameter should reflect the range of values from your data that you want to represent in your report. Data that does not fall within the range of values that is set in the AXIS= parameter will not be represented in the report. You will receive a warning message if all of your data does not fall within the range of values in the AXIS= value list, but the report will continue processing.

For more information about the *value-list*, see the ORDER= parameter in the "AXIS Statement" section of the SAS/GRAPH documentation for your current version of SAS.

Similarly, for more information about the contents of the AXIS statement, see the "AXIS Statement" section of the SAS/GRAPH documentation for your current version of SAS.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **dd:mmm:yyyy:hh:mm:ss**, where **dd** is day, **mmm** is month, **yyyy** is year (in two-digit or four-digit notation), **hh** is hour (using a 24-hour clock), **mm** is minute, and **ss** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BIGYAXIS=YES | NO
specifies whether or not to increase the length of the axis and the font size that is used for the tick marks on the axis. *The default is NO, which means do not increase the axis length or font.* To increase the length of the axis and the font size, specify BIGYAXIS=YES.

BY=*BY-variable-list*
lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three

machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

CL_LAB=*label (obsolete; use LABELS=)*
specifies the label for the class variable. The label has a maximum length of 16 characters.

You can use blank characters in the label (but not an all-blank label). You can enclose the label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the label by using this parameter and the LABELS= parameter, then the label that is specified in this parameter is ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

CL_NAM=*variable-name*
specifies the name of the class variable for the report. For each unique value of the class variable, the macro creates a separate portion or group on the graph.

Compare this type of variable to the variable specified by the macro's BY=parameter. The BY variable creates separate graphs for each unique value of the variable. For example, the following combination of parameters produces one report or plot for each unique value of the BY variable MACHINE:

```
cl_nam=diskid,
by=machine
```

Each of the graphs for MACHINE includes all possible values of DISKID, the class variable. If you are producing a plot, then each unique value of DISKID is represented by a different color or plot symbol. If you are producing a chart, then each value of DISKID is represented by a different column or row.

COLORLST=*value-list*
specifies the colors to use for this spectrum plot. *By default, the color list is represented by a blue-to-red color spectrum and the number of colors that are used is based on the number of values that are specified in the PERCENTS= or DATAVALS= parameter.*

Specify the names of the colors that you want to use on your report, such as blue, green, yellow, orange, or red, by using the following guidelines:

- Use one or more blanks to separate the items in the list, but do not enclose the list in quotation marks.
- The colors are used in ascending order, exactly as you list them in this parameter.
- If the CTEXT= parameter is not specified, then the axis color defaults to the first color in the color list. Do not specify the first color (or any color) in the

color list to be the same as your background color, because anything displayed in the same color as the background appears to be missing.

- The first color that is specified will be used as the color for your first data category. Do not specify the first color (or any color) in the color list to be the same as your background color, because anything displayed in the same color as the background appears to be missing.
- You can specify one or two colors more than the number of color values that are specified in the DATAVALS= or PERCENTS= parameter. (The extra colors can be used to distinguish the missing-values category and the highest-value category.) If the number of colors is incorrect, then the macro stops and a message is printed in the SAS log.

For a complete list of available colors, see SAS/GRAPH Colors in the SAS/GRAPH documentation for your current release of SAS.

CTEXT=*color*
indicates the color of the text on a plot. *The default text color is green*. You can specify the background color by using the CBACK= parameter in the SAS GOPTIONS statement. The default background color is device dependent. Refer to the information on your output device in "Using Graphics Devices" in the SAS/GRAPH documentation for your current release of SAS.

If the CTEXT= parameter is blank, then a color specification for the axis color is searched for in this order:

1. the CTEXT= option in a GOPTIONS statement
2. the default, which is the first color in the color list specified in the GOPTIONS COLORS= statement or the default list of colors for the device.

Note: If the color of the text is the same as the color of the background, then the text appears to be missing. △

DATAVALS=*value-list*
specifies the list of numeric values by which to group your data in this report. For example, if you specify a value list of 50 100 150 200, and the values in your data range from less than 50 to greater than 200, then data values that are less than or equal to 50 would fall in the first category and would be represented in the first color in the color spectrum (see the COLORLST= parameter) for this report. Data values greater than 50 and less than or equal to 100 would fall in the second category and would be represented by the next color in the color spectrum for this report, and so on. The last group would represent data values greater than 200 and would be represented by the last color in the color spectrum for this report.

You can specify a maximum of 13 values. Use blanks to separate the values in the list, but do not enclose the list in quotation marks. You can specify this parameter or the PERCENTS= parameter, but not both. *If you do not specify this parameter, then the values in the PERCENTS= parameter are used. If you do not specify values for the PERCENTS= parameter, then the default values for that parameter are used*.

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*

lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see “Global and Local Macro Variables” on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are

directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file

expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myiurl=http://www.reporter.com/reports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **`&`** obtains the value of the

macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)

specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*

specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their

Web browsers, they see a larger version of the graph report. The larger version is static.

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).

- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LGNDFMT=*format-name* | DATA

specifies the SAS format that is to be used for labels in the legend of your report.

The labels are obtained from the values that are specified for either the PERCENTS= or DATAVALS= parameter. The labels are then listed in the legend of your report by using the format that is specified in the LGNDFMT= parameter.

The value for this parameter can be

- *format-name*, which enables you to specify the name of the format that will be used
- DATA, which causes the labels to be displayed by using the format that is specified for the first analysis variable in the original data set.

SAS IT Resource Management follows these steps to determine which display format to use for the legend:

1. The *format-name* is used, if it is specified.

2 If DATA is specified, then the format specified in the FORMATS= parameter for that variable is used.

3 If the FORMATS= parameter is not specified, then the format that is in the data dictionary for that variable is used.

4 If a format for that variable is not in the data dictionary, then the format defaults to BEST8.

Note: If the formatted width of the data exceeds eight characters, then the BEST8. format is used for that label. △

LTCUTOFF=*time-of-cutoff*
specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

□ *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

□ *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:
If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

OUTDESC=*output-description*
if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*
for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT:* WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename | entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT*: G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see “Examples of the OUT*= Parameters” on page 560.

PALETTE=*palette-name*
specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **`sasuser.palette.win`** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section “Specifying/ Editing/Viewing the Default Palette Definition” in the chapter “Reporting: Working with Palette Definitions” in the *SAS IT Resource Management User’s Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PERCENTS=*value-list*
specifies a list of values (percents without the % signs) by which to group your data for this report. You can specify a maximum of 12 values (which creates 13 range categories) or groups. Analysis variables display in percentile groups and the groups are represented by color (see the COLORLST= parameter) on the graph. *The default value list is 25 50 75 95 99*.

Use one or more blanks to separate the values in the list, but do not enclose the list in quotes. You can specify this parameter or the DATAVALS= parameter, but not both.

PERIOD=*time-interval*
is the summarization period for the report. *The default is ASIS*. For values other than ASIS, the values for the time period within a class or category are combined (as specified in the STAT= parameter) to form a single point on a plot, a bar on a chart, or a row or column in a table.

The following list provides the period name, the effect that it has on the report, and an example of how to specify the period.

- ASIS - leaves datetime in its present form. Example: 09Jun2003:14:37:25
- 15MIN - transforms the time to the first minute of the 15-minute period in the hour and retains the date. Example: 09Jun2003:14:30
- HOUR - transforms the time to the first minute of the hour and retains the date. Example: 09Jun2003:14:00
- 24HOUR - transforms the time to its hour component and omits the date. Range: 0–23. Example: 14
- DATE - omits the time and retains the date. Example: 09Jun2003
- WHOLEDAY - obsolete; use DATE.
- WEEKDAY - transforms the day to the day of the week (where 1=Sunday, 2=Monday, and so on) and omits the month, year, and time. Range: 1–7, which displays as Sunday through Saturday. Example: Monday
- MONTHDAY - omits everything but the day of the month. Range: 1–31. Example: 9
- YEARDAY - transforms the day to the day of the year and omits the month, year, and time. Range: 1–366. Example: 160
- WEEK - transforms the date to the first day of the week as defined by Start-of-Week and omits the time. (Start-of-Week is a PDB property that you can specify with the %CPPDBOPT macro. By default, Start-of-Week is Sunday.) Example: 08Jun2003
- MONTH - omits the day and the time. Example: Jun2003
- YEARMONTH - omits everything but the month of the year. Range: 1–12. Example: 6
- QTR - transforms the month to the first month of the quarter and omits the day and time. Example: Apr2003

Note: You can change the width of the bar that appears on the 3-dimensional plot generated by %CPSPEC. If you specify PERIOD= ASIS, or 15MIN, or HOUR, the bar will be thinner in order to accommodate the increased number of points per line that is typical with shorter periods. △

PROCOPT=*string*
where *string* is any text that is permitted on the PROC statement of the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro. This allows expert SAS/GRAPH users to take advantage of options that are not supported by SAS IT Resource Management macros or new options for which SAS IT Resource Management has not yet added support.

For %CPPLOT1, %CPPLOT2, and %CPSPEC, the corresponding procedure in SAS/GRAPH is PROC GPLOT.

For %CPCCHRT and CPCHART, the corresponding procedure in SAS/GRAPH is PROC GCHART.

For %CPG3D, the corresponding procedure in SAS/GRAPH is PROC G3D.

For %CPTABRPT, the corresponding procedure in the SAS System is PROC TABULATE.

For example, you might specify an annotate data set:

```
%CPPLOT1(....,
         PROCOPT=ANNO=WORK.MYANNO,
         ....);
```

The default value is blank (no value).

For more details about what options are valid on the PROC GPLOT, PROC G3D, and PROC GCHART statements, see your SAS/GRAPH documentation. For information about PROC TABULATE, see your SAS System documentation.

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

STAT=*statistic*
specifies what statistic to use in order to summarize data that spans the time interval that is specified by PERIOD=. The selected statistic is used for all variables in the report. *The default statistic is MEAN*. The STAT= parameter is not valid if PERIOD=ASIS. Valid values for STAT= are as follows:

CSS
is the sum of squares, corrected for the mean.

CV
is the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAX

is the maximum value for all observations.

MEAN
is the arithmetic average. Mean is one measure that is used to describe the center of a distribution of values.

MIN
is the minimum value for all observations.

N
(or COUNT) is the number of observations with nonmissing values for the variables that are being summarized.

NMISS
the number of observations with missing values for the variable being summarized.

RANGE
is the maximum of the difference between the maximum and minimum values for the variables, RANGE=MAX-MIN.

STD
is the standard deviation or square root of variance. Like the variance, it is a measure of the dispersion about the mean, but in the same unit of measure as the data.

STDERR
is the positive square root of the variance of a statistic.

SUM
is the sum of values for all observations.

USS
is the uncorrected sum of squares.

VAR
is a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: If you specify the BY= parameter, the statistics are calculated separately for each value of the BY variable or for each unique combination of the BY variable values, if you specify multiple BY variables. △

STMTOPT=*string*
where *string* is any text that is permitted as an option for the statement in the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro.

For %CPPLOT1 and %CPPLOT2, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPCCHRT and %CPCHART, the statement corresponds to the value of the TYPE parameter. For example, if TYPE=HBAR, then the statement will be the HBAR statement in PROC GCHART, and *string* will be added after the '/' in that statement.

For %CPSPEC, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPG3D, the corresponding statement in SAS/GRAPH is the SCATTER statement in PROC G3D.

For %CPTABRPT, the corresponding statement in SAS is the TABLE statement in PROC TABULATE.

If you want to customize the labeling and details of the axis for the group variable, you can specify an axis statement such as AXIS3 and then instruct %CPCHART to use it:

```
AXIS3 LABEL=(COLOR=BLUE);
%CPCHART(table,
         ....
         GROUP=gvar,
         STMTOPT=GAXIS=AXIS3,
         ....);
```

The default value is blank (no value).

For more details about what options are available for these statements, see the information about PROC GCHART, PROC G3D, and PROC GPLOT in your SAS/GRAPH documentation and PROC TABULATE in your SAS System documentation.

TABTYPE=INTERVAL | EVENT
specifies whether each observation in the data that is read into the table represents a specified interval of time or whether each observation was logged in response to an event, such as when a job began or ended. *If the data is in a view or data set in the detail, day, week, month, or year level of the active PDB, then the default is the value that is specified in the table's definition in the active PDB's data dictionary.*

INTERVAL
each observation represents an interval of time

EVENT
each observation represents an event.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WEIGHT=*weight-variable*
specifies the alternate variable by which to weight (multiply) statistical calculations. If no WEIGHT= variable is specified and the table type is INTERVAL, then the variable DURATION is used as the weight.

For example, if most observations have a value of 15 for the weight variable and one observation has a value of 30 for the weight variable, then that observation has twice the weight of each of the other observations in any calculations.

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

YVAL=*number*
is the number of Y values to print on one page of the plot. *The default number is based on the output device type*. If the number of analysis variables is greater than the value of YVAL=, then the chart is broken into multiple pages.

XVAR=*variable*
specifies the name of the variable for the X axis of a plot. The variable must be numeric. If you do not provide a variable's name, then the variable's name defaults to DATETIME.

%CPSPEC Notes

All variables for the plot should have the same unit of measurement.

Unless otherwise specified by means of the STAT= parameter, each row of the graph plots the mean of the analysis variable.

For more information on the underlying procedure, refer to the GPLOT procedure in the SAS/GRAPH documentation for your current release of SAS.

You can use this macro to produce two styles of reports:

- %CPSPEC Style 1
  - A separate row is generated for each value of the class variable. The row plots the mean of the analysis variable. The color of each symbol corresponds to the value of the mean.
  - The variable DATETIME corresponds to the X axis.
  - You can specify one analysis variable and one class variable.
- %CPSPEC Style 2
  - A separate row is generated for each analysis variable. The row plots the mean of the analysis variable. The color of each symbol corresponds to the value of the mean.
  - The variable DATETIME corresponds to the X axis.
  - You can specify one to fifteen analysis variables.
- For both report styles, you can specify *n* optional BY variables. No BY variables are required. If there is one BY variable, then a separate section of the report is generated for each value of the BY variable. If there is more than one BY variable, then a separate section of the report is generated for each unique combination of values of the BY variables.
- For both report styles, you can specify one optional statistic. The statistic is used to collapse (summarize and replace) data if the Summary Time Period is not AS IS. The default statistic is MEAN.
- For both report styles, you can specify one optional weight variable. In a table of type interval, if no alternate weight variable is specified, DURATION is used as the weight variable.

%CPSRCRPT

Creates a SAS data set and then runs a predefined SAS program against the data set

%CPSRCRPT Overview

The %CPSRCRPT macro enables you to run a complex reporting program that might require multiple steps in order to manipulate the data, while still taking advantage of the %CPRUNRPT macro and the standard SAS IT Resource Management reporting parameters (BEGIN=, BY=, and so on) that are available to supplied report macros. The program can consist of source code from a supplied or custom report definition that you saved through a SAS IT Resource Management GUI, source code that uses SAS DATA step code, and/or source code that runs SAS procedures.

This macro operates as follows:

1. The macro reads an existing data set, manipulates the data (based on some of the parameters of this macro), and then writes another data set.
2. Then, the macro submits the reporting program, with access to all of the parameters that are specified in the %CPSRCRPT macro. Typically, the reporting program reads the newly created data set, does additional manipulation of the data, and writes a graph or text report.

For more specific information, see %CPSRCRPT Notes.

%CPSRCRPT Syntax

%CPSRCRPT(
dataset
,*outdata*
,*rptname*
<,*p4,...,p31*>
<,**BEGIN=***SAS-datetime-value*>
<,**BY=***BY-variable-list*>
<,**CL_NAM=***variable-name*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**OUTTYPE=**GRAPHICS | ASCII>
<,**PALETTE=***palette-name*>
<,**PERIOD=***time-interval*>
<,**REDLVL=***PDB-level*>
<,**SMALLDEV=***device-driver*>
<,**WEBSTYLE=***style*>
<,**WHERE=***where-expression*>);

Details

dataset
specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

outdata
is an obsolete parameter. In the past, the intermediate data was written to and read from &outdata.

Now the intermediate data is written to and read from *&dataset*.

1. %CPSRCRPT macro reads the data set that is specified by the parameter named *dataset*.
2. %CPSRCRPT modifies the data, as specified by the REDLVL=, BEGIN=, BY=, CL_NAM=, END=, FORMATS=, LABELS=, LTCUTOFF=, PERIOD=, and WHERE= parameters. For example, any summarization that is indicated by the PERIOD value is performed, and any subsetting appropriate to the WHERE parameter is done.
3. %CPSRCRPT sets *&dataset* to the location of the modified data (and sets *&outdata* equal to *&dataset*, except in the case of OUTMODE=WEB.)
4. The reporting program (*rptname*) reads the modified data in &dataset as its input data.

rptname
is the four-level name (*libref.catalog_name.entry_name*.SOURCE) of the SAS catalog entry that contains the source code (reporting program) that produces your report when this entry is run by %CPSRCRPT. The libref must be defined by the time that this macro is called. *This positional parameter is required*.

One way to load the source code is to use the %CPCAT macro. (For more information, see "%CPCAT" on page 75.) Another way to load the source code is to enter your code in the SAS Program Editor window, and then to save the code to a catalog by issuing the SAVE command.

p4,...,p31
identifies values that you want to use in your source code. *These positional parameters are optional*, but when they are used, the items must be specified in positions 4 through 31. The values are not used directly by %CPSRCRPT, but can be requested (using *&parameter_name*) by the reporting program (*rptname*) that %CPSRCRPT submits.

You must separate items in the list with commas, but you do not need to use commas as placeholders if you do not use all 31 of these positional parameters. For example, if you use four positional parameters (p4–p7), then you would separate the four parameters with commas, but you do not need to include the additional commas for parameters p8 through p31. For an example of how to use these parameters, refer to the "%CPSRCRPT Examples" section.

The parameters that are specified here should be referenced in your source code as macro variables with names such as

```
&p4
```

and

```
&p5
```

.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **dd:mmm:yyyy:hh:mm:ss**, where **dd** is day, **mmm** is month, **yyyy** is year (in two-digit or four-digit notation), **hh** is hour (using a 24-hour clock), **mm** is minute, and **ss** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **dd:mmm:yyyy** AT **hh:mm:ss**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **hh:mm:ss**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **hh:mm:ss** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).
- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BY=*BY-variable-list*
lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

CL_NAM=*variable-name*
specifies a list of names of variables to use as class variables. If there is more than one class variable, separate each class variable from the next by using one or more blanks. You can specify a maximum of five variables.

If the PERIOD= parameter is not set to ASIS, then a separate observation will be created for each value of the CL_NAM= variable. That is, for each period, there will be one observation for each unique set of values of the class variables.

Note: Although the observations will exist, they will not be sorted. If you want the observations to be sorted, your reporting program must sort them. For example, you could run PROC SORT on the data in *outdata* and use a BY statement of

```
BY=&cl_nam ;
```

△

Note: In the SAS IT Resource Management client GUI, you can specify more than one class variable, but only the first one is used. △

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*
lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see “Global and Local Macro Variables” on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are

directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*

specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **`welcome.htm`**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**`welcome.htm`** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **`HTMLURL=http://www.reporter.com/reports`** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhurl=http://www.reporter.com/reports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **`&`** obtains the value of the macro variable, and the **`%str()`** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file

expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myiurl=http://www.reporter.com/reports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **`&`** obtains the value of the

macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.

- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)

specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*

specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their

Web browsers, they see a larger version of the graph report. The larger version is static.

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).

- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

  ```
  options
     appletloc="../path";
  ```

  Or submit this SAS code on Windows:

  ```
  options
     appletloc="..\path";
  ```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*
specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:
If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.
If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.
(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

OUTDESC=*output-description*
if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is

stored with the report in the catalog that is specified in the OUTLOC= parameter.
If OUTMODE=LPCAT or OUTMODE=GRAPHCAT,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*
for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT:* WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT:* This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTNAME=*filename* | *entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT*: G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)

- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see "Report Name" at the end of the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*. △

For more information about when and how to use the OUTNAME= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTTYPE=GRAPHICS | ASCII
specifies the type of output that the report will create. OUTTYPE= ASCII indicates that the output will be a text-based report; OUTTYPE=GRAPHICS indicates that the output will be a graph.

The default value is **GRAPHICS**.

PALETTE=*palette-name*
specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **sasuser.palette.win** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX

or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/ Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PERIOD=*time-interval*

is the summarization period for the report. *The default is ASIS*. For values other than ASIS, the values for the time period within a class or category are combined (as specified in the STAT= parameter) to form a single point on a plot, a bar on a chart, or a row or column in a table.

The following list provides the period name, the effect that it has on the report, and an example of how to specify the period.

- ASIS - leaves datetime in its present form. Example: 09Jun2003:14:37:25
- 15MIN - transforms the time to the first minute of the 15-minute period in the hour and retains the date. Example: 09Jun2003:14:30
- HOUR - transforms the time to the first minute of the hour and retains the date. Example: 09Jun2003:14:00
- 24HOUR - transforms the time to its hour component and omits the date. Range: 0–23. Example: 14
- DATE - omits the time and retains the date. Example: 09Jun2003
- WHOLEDAY - obsolete; use DATE.
- WEEKDAY - transforms the day to the day of the week (where 1=Sunday, 2=Monday, and so on) and omits the month, year, and time. Range: 1–7, which displays as Sunday through Saturday. Example: Monday
- MONTHDAY - omits everything but the day of the month. Range: 1–31. Example: 9
- YEARDAY - transforms the day to the day of the year and omits the month, year, and time. Range: 1–366. Example: 160

- WEEK - transforms the date to the first day of the week as defined by Start-of-Week and omits the time. (Start-of-Week is a PDB property that you can specify with the %CPPDBOPT macro. By default, Start-of-Week is Sunday.) Example: 08Jun2003
- MONTH - omits the day and the time. Example: Jun2003
- YEARMONTH - omits everything but the month of the year. Range: 1–12. Example: 6
- QTR - transforms the month to the first month of the quarter and omits the day and time. Example: Apr2003

Note: You can change the width of the bar that appears on the 3-dimensional plot generated by %CPSPEC. If you specify PERIOD= ASIS, or 15MIN, or HOUR, the bar will be thinner in order to accommodate the increased number of points per line that is typical with shorter periods. △

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △
- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
 where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

%CPSRCRPT Notes

The macro

1 reads the existing data set, whose location is specified by the *dataset* parameter and REDLVL= parameter.

2 manipulates the data as specified by the following parameters: BEGIN=, BY=, CL_NAM=, END=, FORMATS=, LABELS=, LTCUTOFF=, PERIOD=, and WHERE=.

3 writes the results to another data set, whose location is specified in the macro variable named *DATASET*. (The *outdata* parameter is obsolete.)

4 submits the reporting program, whose location is specified by the *rptname* parameter, and makes available to that program the values of all the parameters that are specified for %CPSRCRPT.

For example, the value of the macro variable *DATASET* is available in the program by referring to **&dataset**, the value of the *p4* parameter is available in the program by referring to **&p4**, the value of the OUTMODE= parameter is available in the program by referring to **&outmode**, and so on.

The reporting program uses SAS procedures and/or SAS DATA step code and/or calls to SAS IT Resource Management macros, and provides to them any relevant values of %CPSRCRPT parameters. Typically, the reporting program

1 reads the data in the newly created data set, by referring to &dataset.

2 might further manipulate the data by means of SAS DATA step code and/or SAS procedures such as PROC SORT and PROC MEANS, while making use of some of the values of the %CPSRCRPT parameters such as &cl_nam and parameters that relate to output locations, such as &outdesc, &outloc, &outmode, &outname, and &outtype.

3 writes one or more graph or text reports by using SAS DATA step code, SAS procedures (PROC GPLOT, PROC G3D, PROC ANOVA, and so on) that produce graph reports, SAS procedures (PROC REPORT, PROC TABULATE, and so on) that produce text reports, and/or SAS IT Resource Management macro calls (%CPCHART, %CPPRINT, and so on) that produce graph or text reports.

- The contents, descriptions, and data format are typically controlled by making use of parameter values such as &p4, ..., &p31, &by, and &cl_nam.
- The destination and output format are typically specified by making use of parameter values such as &outdesc, &outloc, &outmode, &outname, &outtype (and for Web-enabled reports, &htmldir, &htmlurl, &imagedir, &imageurl, &largedev, &palette, &smalldev, and &webstyle).

In the reporting program that is specified by the parameter *rptname*:

- For graph reports:
  - if OUTMODE= is not WEB, use &dataset, &by, &outmode, &outloc, &outname, and &outdesc in the locations shown in the following:
    - This pseudo code for a SAS graphic procedure:

```
PROC whatever ... DATA=&dataset GOUT=&outloc
;
   BY &by;
   PLOT | HBAR | SCATTER | etc. statement ;
   NAME="&outname" DES="&outdesc" ;
```

    - This pseudo code for a call to a SAS IT Resource Management macro:

```
%CPxxxxxx(&dataset
          ...
          ,by=&by
          ...
          ,outmode=&outmode
```

```
          ,outloc=&outloc
          ,outname=&outname
          ,outdesc=&outdesc
          ... );
```

- if OUTMODE=WEB, use &dataset, &by, &outloc, and &outdesc in the locations shown in the following:
  - This pseudo code for a SAS graphics procedure:

```
%if &by eq %str()
   %then %let bystmt = BY _ID ;
%else %let bystmt = BY _ID &by ;

PROC whatever ... DATA=&dataset GOUT=&outloc ;
   &bystmt ;
   PLOT | HBAR | SCATTER | etc. statement ;
      NAME="_0000001" DES="&outdesc" ;
```

  - This pseudo code for a call to a SAS IT Resource Management macro:

```
%CPxxxxxx(&dataset
          ...
          ,by=&by
          ...
          ,outmode=&outmode
          ,outloc=&outloc
          ,outname=&outname
          ,outdesc=&outdesc
          ,htmldir=&htmldir
          ... );
```

For examples of graph reports that are produced by SAS graphics procedures, see Examples 1 through 3 below. For an example of a graph report that is produced by a call to a SAS IT Resource Management macro, see Example 5 below.

- For text reports:
  - if OUTMODE= is not equal to WEB, use &dataset, &by, &outmode, &outloc, &outname, and &outdesc in the locations shown in the following:
    - This pseudo code for a SAS text procedure:

```
&prtto
    ...
    PROC whatever DATA=&dataset;
       %if &by ne &str() %then %str(BY &by;);
    ...
&prttend
```

    - This pseudo code for a call to a SAS IT Resource Management macro:

```
%CPxxxxxx(&dataset
          ...
          ,by=&by
          ...
          ,outmode=&outmode
          ,outloc=&outloc
          ,outname=&outname
          ,outdesc=&outdesc
```

```
            ... );
```

- if OUTMODE=WEB, use &dataset, &by, &outmode, &outloc, and &outname in the locations shown in the following:
  - This pseudo code for a SAS text procedure:

```
&prtto
    ...
    PROC whatever DATA=&dataset;
        %if &by ne &str() %then %str(BY &by;);
    ...
&prttend
```

  - This pseudo code for a call to a SAS IT Resource Management macro:

```
%CPxxxxxx(&dataset
          ...
          ,by=&by
          ...
          ,outmode=&outmode
          ,outloc=&outloc
          ,outname=&outname
          ,outdesc=&outdesc
          ,htmldir=&htmldir
          ... );
```

For an example of a text report that is produced by a SAS text procedure, see Example 4 below.

In the call to %CPSRCRPT:

- For graph reports:
  - if OUTMODE= is not WEB, provide values for OUTMODE=, OUTLOC=, OUTNAME=, and OUTDESC=, as described in "How the OUT*= Parameters Work Together" on page 551. Do not specify OUTTYPE=.
  - if OUTMODE=WEB, provide values for OUTMODE=, OUTLOC=, OUTNAME=, OUTDESC=, and HTMLDIR= as described in "How the OUT*= Parameters Work Together" on page 551. Do not specify OUTTYPE=.
- For text reports:
  - if OUTMODE= is not WEB, provide values for OUTMODE=, OUTLOC=, OUTNAME=, and OUTDESC=, as described in "How the OUT*= Parameters Work Together" on page 551. Also specify OUTTYPE=ASCII.
  - if OUTMODE=WEB, provide values for OUTMODE=, OUTLOC=, OUTNAME=, OUTDESC=, and HTMLDIR= as described in "How the OUT*= Parameters Work Together" on page 551. Also specify OUTTYPE=ASCII.

Note: In the SAS IT Resource Management GUI for z/OS, you can select a supplied %CPSRCRPT report definition and run it, but you cannot create your own %CPSRCRPT report definition or modify an existing %CPSRCRPT report definition. In the SAS IT Resource Management GUI for UNIX and Windows, you can select, run, create, and modify a %CPSRCRPT report definition. In batch mode on all platforms, you can run existing %CPSRCRPT report definitions. △

%CPSRCRPT Examples

Each example has two sections. The first section has a description about and code for *storing* the reporting program. The second section has a description about and code for

running the reporting program. The code actually continues from the first section to the second section. The code is displayed in two parts only to make the descriptions easier to compare with the code.

Example 1

This example uses a SAS graph procedure and produces a graph report, but does not access the %CPSRCRPT parameters and cannot direct the report to the Web.

To store the reporting program:

In the code below, the %CPSTART macro is called to start SAS IT Resource Management, so that the %CPCAT macro will be recognized.

The first call to the %CPCAT macro reads the reporting program. The program does not run now, but this is what it will do when %CPSRCRPT submits it later:

- The reporting program plots the values of variables IFIOCTS and DATETIME from the intermediate data set whose name is specified by &dataset and stores the resulting graph in the SAS catalog WORK.MYWORK.

  Notice that this code ignores the %CPSRCRPT output parameters, and has the output hardcoded (catalog GOUT=WORK.MYWORK, entry NAME=INOCTS, description DES=My Important Graph) to match the values in the call to %CPSRCRPT. (Example 2 shows a non-hardcoded version.)

The second call to the %CPCAT macro stores the reporting program in an entry named TMP1.SOURCE in the SAS catalog named WORK.MYPGMS.

```
%CPSTART(
pdb="!sasroot/cpe/pdb-hpo", mode=batch );
%cpcat(); cards4;
proc gplot data=&dataset gout=work.mywork ;
     title 'Input Octets' ;
     plot ifiocts * datetime = "+"
        name = inocts
        des = My Important Graph ;
     run ;
;;;;

%cpcat(cat=work.mypgms.tmp1.source);
```

To run the reporting program:

In the code below, the call to the %CPSRCRPT macro reads the data in DETAIL.HN2IFT, subsets it by using the BEGIN=, END=, and WHERE= parameters, summarizes it as specified by the PERIOD= parameter, and writes the data to a data set whose name is stored in the macro variable named DATASET.

Then, the %CPSRCRPT macro submits the WORK.MYPGMS.TMP1.SOURCE program to SAS.

```
%CPSTART( pdb="!sasroot/cpe/pdb-hpo", mode=batch );
%CPSRCRPT  ( hn2ift,
             ,
             work.mypgms.tmp1.source,
             redlvl=detail,
             period=24hour,
             outmode=graphcat,
             outloc=work.mywork,
```

```
outname=inocts,
outdesc=My Important Graph,
begin=TODAY-7,
end=TODAY-1,
where=machine='bigapple.net.sas.com' );
```

Example 2

This example uses a SAS graph procedure, produces a graph report, makes use of the %CPSRCRPT parameters, and cannot direct the report to the Web. (This example builds on Example 1.)

To store the reporting program:

In the code below, the %CPSTART macro is called to start SAS IT Resource Management.

The first call to the %CPCAT macro reads the reporting program. The program does not run now, but this is what it will do when %CPSRCRPT submits it later:

- The program, which is a variation of the one in Example 1 in order to illustrate how to access values of the %CPSRCRPT parameters rather than use hardcoded values, creates a macro called TEMP and then runs the TEMP macro.

  The TEMP macro runs PROC GPLOT on the data in &dataset and writes the resulting graph to entry INOCTS (with description My Important Graph) in catalog WORK.MYWORK, as specified by the OUTMODE=, OUTLOC=, OUTNAME=, and OUTDESC= parameters.

The second call to the %CPCAT macro stores the reporting program in an entry named TMP2.SOURCE in the SAS catalog named WORK.MYPGMS.

```
%CPSTART( pdb="!sasroot/cpe/pdb-hpo", mode=batch );
%CPCAT ; cards4;

%macro temp();

/* if outloc was specified, puts it into the */
/* right syntax for use on PROC GPLOT stmt   */
%if &outloc ne %str()
   %then %let rptout = GOUT=&outloc ;

/* if outname was specified, puts it into the */
/* right syntax for use on PLOT stmt          */
%if &outname ne %str()
   %then %let rptnm = NAME="&outname" ;

/* uses an '*' if user doesn't */
/* provide a character         */
%if &p4 eq %str()
   %then %let p4 = * ;

/* uses description if user specified one */
%if &outdesc ne %str()
   %then %let desc = DES="&outdesc";

/* adds GOUT=&outloc to GPLOT statement */
proc gplot data=&dataset &rptout ;
```

```
     title 'Input Octets';

     /* uses plot character if given */
     plot ifiocts * datetime = "&p4"

          /* uses name if given */
          &rptnm

          /* uses description if given */
          &desc ;

     run;

%mend temp;
%temp();

;;;;
%CPCAT( cat=work.mypgms.tmp2.source) ;
```

To run the reporting program:

In the code below, the call to the %CPSRCRPT macro reads the data in DETAIL.HN2IFT, subsets the data as specified by the BEGIN=, END=, and WHERE= parameters, summarizes the data as specified by the PERIOD= parameter, and writes the data to a data set whose name is stored in the macro variable named &dataset.

Then, %CPSRCRPT submits the reporting program in WORK.MYPGMS.TMP2.SOURCE.

```
%CPSTART( pdb="!sasroot/cpe/pdb-hpo", mode=batch );
%CPSRCRPT  ( hn2ift,
             ,
             work.mypgms.tmp2.source,
             +,         /* want plot char to be a '+' */
             redlvl=detail,
             period=24hour,
             outmode=graphcat,
             outloc=work.mywork,
             outname=inocts,
             outdesc=My Important Graph,
             begin=TODAY-7,
             end=TODAY-1,
             where=machine='bigapple.net.sas.com' );
```

Example 3

This example uses a SAS graph procedure, produces a graph report, accesses the %CPSRCRPT parameters, and does direct the report to the Web. (This example builds on Example 2.)

To store the reporting program:

In the code below, the %CPSTART macro is called to start SAS IT Resource Management.

The first call to the %CPCAT macro reads the reporting program. The program does not run now, but this is what it will do when %CPSRCRPT submits it later:

- The program, which is a variation of the one in Example 2 in order to illustrate how to generate a Web-enabled report, creates a macro called TEMP that then runs PROC GPLOT on the data in &dataset, and first writes the resulting graph to the OUTLOC= catalog's entry _0000001, or higher if there is already such an

entry. The entry has the description My Important Graph and is in catalog WORK.MYWORK as specified by the OUTLOC=, OUTNAME=, and OUTDESC= parameters.

The second call to the %CPCAT macro stores the reporting program in an entry named TMP3.SOURCE in the SAS catalog named WORK.MYPGMS.

```
%CPSTART( pdb="!sasroot/cpe/pdb-hpo", mode=batch );
%CPCAT ; cards4;

%macro temp;

/* if outloc was specified, puts it into the */
/* right syntax for use on PROC GPLOT stmt   */
%if &outloc ne %str() and (&outmode=GRAPHCAT or &outmode=WEB)
   %then %let rptout = GOUT=&outloc ;

/* if outname was specified, puts it into   */
/* the right syntax for use on GPLOT stmt   */
/* also makes a BY statement if appropriate  */
%if &outmode ne WEB and &outname ne %str()
   %then %let rptnm = NAME="&outname" ;
%else %if &outmode eq WEB
   %then %do;
      %let rptnm = NAME="_0000001";
      %if &by eq %str()
         %then %let bystmt = BY _ID ;
      %else %let bystmt = BY _ID &by ;
      %end;
%else %let rptnm = %str();

/* uses an '*' if user doesn't */
/* specify a plot character    */
%if %quote(&p4) eq %str()
   %then %let p4 = * ;

/* if outdesc was specified, puts */
/* it into the right syntax       */
%if &outdesc ne %str()
   %then %let desc = des="&outdesc" ;

/* adds GOUT=&outloc to GPLOT statement  */
proc gplot data=&dataset &rptout ;

     title 'Input Octets';

     /* uses BY list or not */
     &bystmt ;

     /* uses plot char if given */
     plot ifiocts * datetime = "&p4" /

        /* uses name or _0000001 */
        &rptnm
```

```
        /* uses description if given */
        &desc ;

     run;

%mend temp;
%temp;

;;;;
%CPCAT( cat=work.mypgms.tmp3.source) ;
```

To run the reporting program:

In the code below, the call to the %CPSRCRPT macro reads the data in DETAIL.HN2IFT, subsets the data as specified by the BEGIN=, END=, and WHERE= parameters, summarizes the data as specified by the PERIOD= parameter, and writes the data to a data set whose name is stored in the macro variable named &dataset.

Next, %CPSRCRPT submits the reporting program in WORK.MYPGMS.TMP3.SOURCE.

Then, because OUTMODE=WEB, %CPSRCRPT goes on to write the graph to the directory c:\webtest\test (as specified by the HTMLDIR= parameter) in the report group My Important Graph as specified by the OUTDESC= parameter.

```
%CPSRCRPT  ( hn2ift,
             ,
             work.mypgms.tmp3.source,
             +,         /* want plot char to be a '+' */
             redlvl=detail,
             period=24hour,
             outmode=web,
             outloc=work.mywork,
             outdesc=My Important Graph,
             htmldir=c:\webtest\test,
             begin=latest-1,
             end=latest-1,
             where=machine='bigapple.net.sas.com' );
```

Example 4

This example uses a SAS text procedure, produces a text report, accesses the %CPSRCRPT parameters, and directs the report to the Web.

To store the reporting program:

In the code below, the %CPSTART macro is called to start SAS IT Resource Management.

The first call to the %CPCAT macro reads the reporting program. The program does not run now, but this is what it will do when %CPSRCRPT submits it later:

- The program, which is similar to the one in Example 3 in order to illustrate how to generate a Web-enabled text report, runs the SAS PRINT procedure on the data in &dataset and writes the printout to the file SRCTEST1.htm (as specified by the OUTNAME= parameter) in the directory named c:\webtest\test1 (as specified by the HTMLDIR= parameter). The report is in the report group named CPSRCRPT Text1 (as specified by the OUTDESC= parameter).

The second call to the %CPCAT macro stores the reporting program in an entry named TMP4.SOURCE in the SAS catalog named WORK.MYPGMS.

```
%CPSTART( pdb="!sasroot/cpe/pdb-hpo", mode=batch );
%cpcat(); cards4;
  &prtto
  Title 'Print of detail-level HN2IFT table';
  proc print data=&dataset;
     var machine udate uweek ifouctp iftype ;
  run;
  &prttend
;;;;
%cpcat(cat=work.mypgms.tmp4.source);
```

To run the reporting program:

In the code below, the call to the %CPSRCRPT macro reads the data from DETAIL.HN2IFT, subsets the data as specified by the BEGIN=, END=, and WHERE= parameters, summarizes the data as specified by the PERIOD= parameter, and writes the data to a data set whose name is stored in the macro variable named &dataset.

Then, the %CPSRCRPT macro submits the WORK.MYPGMS.TMP4.SOURCE program to SAS.

```
%cpsrcrpt(hn2ift,
          ,
          work.mypgms.tmp4.source,
          redlvl=detail,
          period=24hour,
          outmode=web,
          outloc=mytexts,
          outtype=ASCII,
          htmldir=c:\webtest\test1,
          outdesc=CPSRCRPT Text1,
          outname=SRCTEST1,
          begin=latest-1,
          end=latest-1,
          where=machine='bigapple.net.sas.com');
```

Example 5

This example uses a SAS IT Resource Management macro, produces a graph report, and accesses the %CPSRCRPT parameters, and can direct the report to the Web. Two examples of the call to %CPSRCRPT are shown: one does not make use of the ability to direct the reports to the Web; the other does make use of that ability.

To store the reporting program:

In the code below, the %CPSTART macro is called to start SAS IT Resource Management.

The first call to the %CPCAT macro reads the reporting program. The program does not run now, but this is what it will do when %CPSRCRPT submits it later:

- The program creates a macro called TEMP.

  The macro runs a SAS DATA step, which reads its input from &dataset and writes its output to WORK.RMFINTRV.

  Next, the data is sorted.

  Then, the sorted data is read by the SAS IT Resource Management macro named %CPCHART.

  %CPCHART (as specified by OUTMODE=GRAPHCAT) writes an entry named INOCTS2 (as specified by OUTNAME=) with a description My Macro Graph (as

specified by OUTDESC=) to the SAS catalog named WORK.MYWORK (as specified by OUTLOC=).

The second call to the %CPCAT macro stores the reporting program in an entry named TMP5.SOURCE in the SAS catalog WORK.MYPGMS.

```
%CPSTART( pdb="!sasroot/cpe/pdb-smf", mode=batch );
%CPCAT ; cards4;

%macro temp;

   DATA rmfintrv(keep = date hour interval class
                        pageins shift datetime);
      length class oth0-oth9 $ 15;
      retain oth0-oth9 ' ';
      set &dataset;
      in _N_ = 1 then do;
         call label(ot0pgin,oth0);
         call label(ot1pgin,oth1);
         call label(ot2pgin,oth2);
         call label(ot3pgin,oth3);
         call label(ot4pgin,oth4);
         call label(ot5pgin,oth5);
         call label(ot6pgin,oth6);
         call label(ot7pgin,oth7);
         call label(ot8pgin,oth8);
         call label(ot9pgin,oth9);
         oth0 = scan(oth0,1);
         oth1 = scan(oth1,1);
         oth2 = scan(oth2,1);
         oth3 = scan(oth3,1);
         oth4 = scan(oth4,1);
         oth5 = scan(oth5,1);
         oth6 = scan(oth6,1);
         oth7 = scan(oth7,1);
         oth8 = scan(oth8,1);
         oth9 = scan(oth9,1);
       end;
       nconst=1;
       h = hour(datetime);
       m = minute(datetime);
       interval = put(h,z2.)||put(m,z2.);
       date = datepart(datetime); format date date9.;
       class = 'Overhead';
       pageins = cpopgin*nconst;
       if pageins > 0 then output;
       class = 'TSO';
       pageins = tsopgin*nconst;
       if pageins > 0 then output;
       class = 'IMS';
       pageins = imspgin*nconst;
       if pageins > 0 then output;
       class = 'CICS';
       pageins = cicspgin*nconst;
       if pageins > 0 then output;
       class = 'Batch';
```

```
        pageins = batpgin*nconst;
        if pageins > 0 then output;
        class = 'Other';
        pageins = otrpgin*nconst;
        if pageins > 0 then output;
        class = oth0;
        pageins = ot0pgin*nconst;
        if pageins > 0 then output;
        class = oth1;
        pageins = ot1pgin*nconst;
        if pageins > 0 then output;
        class = oth2;
        pageins = ot2pgin*nconst;
        if pageins > 0 then output;
        class = oth3;
        pageins = ot3pgin*nconst;
        if pageins > 0 then output;
        class = oth4;
        pageins = ot4pgin*nconst;
        if pageins > 0 then output;
        class = oth5;
        pageins = ot5pgin*nconst;
        if pageins > 0 then output;
        class = oth6;
        pageins = ot6pgin*nconst;
        if pageins > 0 then output;
        class = oth7;
        pageins = ot7pgin*nconst;
        if pageins > 0 then output;
        class = oth8;
        pageins = ot8pgin*nconst;
        if pageins > 0 then output;
        class = oth9;
        pageins = ot9pgin*nconst;
        if pageins > 0 then output;
    RUN;

    PROC SORT DATA=rmfintrv;
         by date hour shift;
    RUN;

GOPTIONS ctitle=white ftitle=duplex;
TITLE1 "Page-ins by Performance Group";
%CPCHART(WORK.RMFINTRV,HOUR,HOUR,PAGEINS,PAGE_INS
          ,YVAL=16
          ,OUTMODE=&outmode
          ,OUTLOC=&outloc
          ,OUTNAME=&outname
          ,OUTDESC=&outdesc
          ,HTMLDIR=&htmldir
          ,STAT=SUM
          ,SUBGROUP=CLASS
          ,TYPE=VBAR
          ,REDLVL=OTHER);
```

```
%mend temp;
%temp;

;;;;
%CPCAT( cat=work.mypgms.tmp5.source) ;
```

To run the reporting program:

In the code below, the call to the %CPSRCRPT macro reads the data in DETAIL.XRMFINT, subsets the data as specified by the BEGIN=, END=, and WHERE= parameters, summarizes the data as specified by the PERIOD= parameter, and writes the data to the data set named &dataset.

Then %CPSRCRPT submits the reporting program in WORK.MYPGMS.TMP5.SOURCE. %CPSRCRPT can ignore the reporting program's ability to direct the report to the Web, as shown in the following code:

```
%CPSRCRPT  ( xrmfint,
             ,
             work.mypgms.tmp5.source,
             redlvl=detail,
             period=24hour,
             outmode=graphcat,
             outloc=work.mywork,
             outname=My_rpt,
             outdesc=My Macro Graph,
             begin=latest-1,
             end=latest-1,
             where=machine='os390a.net.sas.com' );
```

Or %CPSRCRPT can take advantage of the reporting program's ability to direct the report to the Web, as shown in this code:

```
%CPSRCRPT  ( xrmfint,
             ,
             work.mypgms.tmp5.source,
             redlvl=detail,
             period=24hour,
             outmode=WEB,
             outloc=work.mywork,
             outdesc=My Macro Graph,
             htmldir=c:\temp\testing,
             begin=latest-1,
             end=latest-1,
             where=machine='os390a.net.sas.com' );
```

%CPTABRPT

Generates tabular reports

%CPTABRPT Overview

The %CPTABRPT macro generates tabular reports by executing the SAS TABULATE procedure. This macro can generate any of six different tabular reports. You can specify a maximum of 10 analysis variables for this report, and the other parameters that you specify depend on your report type.

For more information about the underlying procedure, refer to the *SAS Guide to TABULATE Processing* for your current release of SAS.

%CPTABRPT Syntax

%CPTABRPT(
dataset
,*var1*<,*lab1*><,*fmt1*>...<,*var10*><,*lab10*><,*fmt10*>
,**CL_NAM=***variable-name-list*
,**STAT=***statistic*
<,**ACROSS=***class-variable-list*>
<,**BEGIN=***SAS-datetime-value*>
<,**BOX=***'text'*>
<,**BY=***BY-variable-list*>
<,**CL_LAB=***label-list (obsolete; use LABELS=)*>
<,**END=***SAS-datetime-value*>
<,**FORMATS=**(*var-format-pairs*)>
<,**HTMLDIR=***directory-name* | *PDS-name*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LABELS=**(*var-label-pairs*)>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**MISS=**YES | NO>
<,**MISSTEXT=***'text'*>
<,**NCOLS=***line-size*>
<,**NROWS=***page-size*>
<,**ORDER=**DATA | FORMATTED | FREQ | INTERNAL>
<,**OUTDESC=***output-description*>
<,**OUTLOC=***output-location*>
<,**OUTMODE=***output-format*>
<,**OUTNAME=***physical-filename* | *entry-name*>
<,**PALETTE=***palette-name*>
<,**PROCOPT=***string*>
<,**PRTMISS=**YES | NO>
<,**REDLVL=***PDB-level*>
<,**ROTATE=**LANDSCAPE | PORTRAIT>
<,**RTS=***number*>
<,**SEPS=**YES | NO>
<,**SMALLDEV=***device-driver*>
<,**STATFMT=***format*>
<,**STATLAB=***'label'*>
<,**STMTOPT=***string*>
<,**SUMMARY=**NONE | ALL | LAST>
<,**TABTYPE=**INTERVAL | EVENT>
<,**TYPE=**TYPE1 | TYPE2 | TYPE3 | TYPE4 | TYPE5 | TYPE6>
<,**WEBSTYLE=***style*>
<,**WHERE=***where-expression*>);

Details

dataset

specifies the name of the data set from which to generate the report. *This positional parameter is required.* It can be specified in one of three ways.

If the data is in the detail, day, week, month, or year level of the active PDB, then specify the value of this parameter in one of these ways:

- Specify the table name via the *dataset* parameter and specify the level via the REDLVL= parameter. If the REDLVL= parameter is not specified, then by default REDLVL=DETAIL.
- Specify the view name (*REDLVL_name.TABLE_name*) by using the *dataset* parameter, and do not specify the REDLVL= parameter.

If the data is not in the detail, day, week, month, or year level of the active PDB, then

- specify the data set name (in the format *libref.data-set-name*) by using the *dataset*parameter, and specify REDLVL=OTHER.

Note: When you specify the data set name by using the *libref* format, the libref must be defined before this macro is called. For more information about defining a libref, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

var1,<lab1>,<fmt1> ..., <var10>,<lab10>,<fmt10>

specifies a list of triplets. Each triplet consists of the name of one analysis variable, the label for that analysis variable, and the SAS format for that analysis variable. Each label has a maximum length of 15 characters.

The maximum number of triplets that you can specify is 10.

If you specify multiple analysis variables, then the unit of measure for all the variables should be the same and the range of values should be similar.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or in double quotation marks, but the quotation marks are not required. If the body of a label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in a label.

You are required to specify at least one analysis variable. However, you are not required to specify labels when you use this parameter. In fact, using the LABELS= parameter to specify the labels is preferred. Because these are positional parameters, you must use a comma as a placeholder for a label that you omit. Similarly, you must use a comma as a placeholder for a format that you omit. For example, this code shows two variables, each with its own format, but only one with a label.

```
...,var1,lab1,fmt1,var2,,fmt2,...
```

You can specify the labels by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify the LABELS= parameter, then any labels that are specified by using this parameter are ignored. If labels are not specified by using this parameter and not specified by using the LABELS= parameter, then the variables' labels in the PDB's data dictionary are used.

For more information about SAS formats, refer to "SAS Formats" in the *SAS Language Reference* for your current release of SAS.

CL_NAM=*variable-name–list*

specifies the class variables for this report. A separate row or group of analysis variables is displayed for each unique value of the class variable. *This parameter is required for all report styles except TYPE1.* If there are multiple variables, separate them with one or more blanks. You can specify a maximum of five variables.

STAT=*statistic*

specifies a maximum of five statistics to calculate for the analysis variables in this report. *For %CPTABRPT, MEAN is the default for the TYPE1 report style; for all other report styles except TYPE4, the parameter is required. The parameter is ignored if TYPE=TYPE4.*

You might want to try more than one statistic, such as SUM and MEAN, in order to see which one makes your report easiest to interpret. For information about how to label statistics, see the %CPTABRPT macro's STATLAB parameter.

You can specify a distinct format (see the STATFMT= parameter) for each statistic that you request. However, if you do specify multiple statistics and also a format, then you need to specify a format for each statistic. You can use a '.' for statistics that you want to use a default format. Valid values for STAT= are as follows:

CSS
is the sum of squares, corrected for the mean.

CV
is the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAX
is the maximum value for all observations.

MEAN
is the arithmetic average. Mean is one measure that is used to describe the center of a distribution of values. Other measures include mode and median.

MIN
is the minimum value for all observations.

N
or COUNT, is the number of observations with nonmissing values for the variables that are being summarized.

NMISS
is the number of observations with missing variable values.

RANGE
is the maximum of the difference between the maximum and minimum values for the variable MAX-MIN.

STD
is the standard deviation or square root of variance. Like the variance, it is a measure of the dispersion about the mean, but in the same unit of measure as the data.

STDERR
is the positive square root of the variance of a statistic.

SUM
is the sum of values for all observations.

USS
is the uncorrected sum of squares.

VAR
is a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

ACROSS=*class-variable-list*

specifies the variables to use for column headers. *This parameter is required if TYPE=TYPE5*. You can specify a maximum of five variables. Separate the values in the list with one or more blank spaces.

BEGIN=*SAS-datetime-value*
specifies the beginning datetime of a datetime range that is used to subset the observations. If the beginning date is specified but the beginning time is not specified, then the beginning time defaults to 00:00:00.00. If neither the beginning date nor the beginning time is specified, then the datetime defaults to the value of the variable DATETIME on the oldest observation. *This parameter is optional.*

Note: SAS date formats support both two- and four-digit year values. (The interpretation of a two-digit year depends on the setting of the SAS system option YEARCUTOFF.) △

The datetime value that specifies the beginning or ending of the datetime range can have one of the following syntaxes:

- a valid SAS datetime value, such as **`dd:mmm:yyyy:hh:mm:ss`**, where **`dd`** is day, **`mmm`** is month, **`yyyy`** is year (in two-digit or four-digit notation), **`hh`** is hour (using a 24-hour clock), **`mm`** is minute, and **`ss`** is second.
- a valid SAS date value, followed by AT or @, followed by a valid SAS time value. An example is **`dd:mmm:yyyy`** AT **`hh:mm:ss`**. (Blanks around the @ or AT are optional.)
- a keyword date value, followed by a valid SAS time value. (For more about keyword date values, see the following keyword value descriptions.) An example is LATEST AT **`hh:mm:ss`**.
- a keyword date value, with an offset (in days, weeks, months, or years), followed by a valid SAS time value. For example, you can specify LATEST-2 days AT **`hh:mm:ss`** or you can specify a date such as Today -1 WEEK @ 09:00. If you specify only an offset number without a unit, then the default unit is the unit that is associated with the level of the PDB on which you are reporting.

Note: The value for BEGIN= can use a different syntax from the value for END=. △

The following keywords can be used for BEGIN= and END=:

- EARLIEST means the date of the first (oldest; minimum date) observation for the specified table at the specified level of the PDB. This is based on the observation's value of the variable DATETIME.
- TODAY means the current date when you run the report definition.
- LATEST means, roughly, the date of the last (newest; maximum date) observation. This is based on the observation's value of the variable DATETIME. More exactly, in the case of the LATEST keyword, there is a separate parameter LTCUTOFF= whose time enables a decision about whether LATEST is the date of the last observation or LATEST is the previous day. Note that LTCUTOFF= has nothing to do with the time for subsetting. It affects only the date that is to be used for the value of LATEST.

Default values for BEGIN=, END=, and LTCUTOFF= parameters are as follows:

- Keyword value - BEGIN=EARLIEST, END=LATEST, and LTCUTOFF=00:00:00.
- Unit - the unit that is associated with the level of the PDB on which you are reporting (day, week, month, year). If the level is set to OTHER, then the macro reads the data in order to determine the earliest and most recent datetime values (because there is no table definition).

- Time - BEGIN=00:00 on the specified date and END=23:59:59 on the specified date.

Examples:

- The following combination of values reports on the last three days of data, beginning at 8:00 a.m. three days ago and ending today at 5:00 p.m.:

```
begin=today-3@08:00, end=today at 17:00
```

- The following example reports on the selected level of the PDB (for this report definition). This combination begins at 0:00 three days after the oldest date and ends at 23:59:59 on the date that is two days prior to the maximum date:

```
begin=earliest+3, end=latest-2
```

- The following example changes the unit of measurement by specifying the exact unit. This example includes the most recent two weeks (14 days) of data.

```
begin=latest-2weeks, end=latest
```

- If the newest observation has a DATETIME value of **'12APR2004:04:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **12APR2004**, because **04:30** is beyond the cutoff time of **04:15**.
- If the newest observation has a DATETIME value of **'12APR2004:03:30'dt** and the value of LTCUTOFF= is set to **04:15**, then the value of LATEST becomes **11APR2004**, because **03:30** is not beyond the cutoff time of **04:15**.

BOX='*text*'
places text in the box above the row titles on the left side of the table.

BY=*BY-variable-list*
lists the variables in the order in which you want them sorted for your report. A separate graph (for graph reports) or page (for text reports) is produced for each value of the BY variable or for each unique combination of BY variable values if you have multiple BY variables. For example, if you have four disk IDs on three machines, then the following setting for the BY= parameter produces twelve graphs or pages, one for each of the twelve unique combinations of machine and disk ID values:

```
by=machine diskid
```

A new page (if the OUTMODE= parameter is not equal to WEB, or if OUTMODE=WEB and the value of HTMLDIR= is a PDS) or a new file (if OUTMODE=WEB and the value of HTMLDIR= is a directory or folder) will start when the value of any variable in this list changes. The %CPTABRPT macro checks how many pages or files will be generated. If the number is greater than a limit, then no pages or files are generated, the macro terminates and writes an ERROR message to the log, and the job continues. If you want to change the limit, add this statement before the call to the macro:

```
%let cp_pglim = n ;
```

where *n* is the new limit. By default, *n* is 500.

Suppose that OUTMODE=WEB, the BY= variable is not specified, and the file name is *rptname*.htm. Then, if OUTMODE=WEB and the BY= variable is specified and the report is directed to a folder or directory, the filenames takes the form *rptname*_page*n*.htm. (For example, page 3 of a report named *myrpt* would be *myrpt_page3.htm*.) For more information about report output filenames, see the OUTNAME= parameter.

Note: On z/OS, if the following three conditions are all true, then you might want to have the HTMLDIR= parameter direct the reports to a directory in the UNIX File System area instead of directing the report to a PDS:

- you are calling the %CPTABRPT macro
- the OUTMODE= parameter is set to WEB
- you are using the BY= parameter.

When the report is directed to a PDS, the report is a very large single file without the typical filtering. When the report is directed to a directory, the report becomes multiple small files with typical filtering. △

For an alternate method of handling unique values of variables, refer to the description of class variables.

If there is no BY= parameter, then observations are sorted in the order in which the variables are listed in the level from which the report is obtaining data:

- the BY variables list at the detail level of the specified table in the PDB
- the class variables list in the specified level of the specified table in the PDB.

CL_LAB=*label-list (obsolete; use LABELS=)*
specifies the labels for the class variables that are used in the table. Each label has a maximum length of 16 characters.

Separate multiple labels with blanks. If you specify this parameter, then the number of labels should match the number of class variables. You can use a period to indicate a missing label; here is an example:

```
cl_nam=var1 var2 var3,
cl_lab=lab2 .    lab3
```

As shown in this example, commas separate the parameters, not the labels. Use one or more blanks to separate the labels in the list.

You can use blank characters in a label (but not an all-blank label). You can enclose a label in single quotation marks or double quotation marks, but the quotation marks are not required. If the body of the label contains an unmatched single quotation mark, then use matched double quotation marks to enclose the label, and vice versa. Do not include commas, equal signs, parentheses, or ampersands in the label.

You can specify the label by using this parameter or the LABELS= parameter. Using LABELS= is the preferred method. If you specify labels by using this parameter and the LABELS= parameter, then the labels that are specified in this parameter are ignored. If a label is not specified by using this parameter and not specified by using the LABELS= parameter, then the variable's label in the PDB's data dictionary is used.

END=*SAS-datetime-value*
specifies the ending datetime of a datetime range that is used to subset the observations. If you are subsetting both by WHERE and the datetime range is specified, then the subset of observations that are used satisfies both criteria.

This parameter is optional.

Use the END= parameter in combination with BEGIN= in order to define the range for subsetting data for the report. For additional information and examples, see the BEGIN= parameter.

FORMATS=*(var-format-pairs)*
lists the names of variables that are used in this report definition and the format to use for each variable. You must enclose the list in parentheses and use at least one space between each variable format and the next variable name in the list. Do not enclose the values in quotes. Here is an example:

```
formats=(cpubusy=best8. machine=$32.)
```

These variable formats are stored with this report definition but are not stored in the data dictionary. If you do not specify a format for a variable, then the format for that variable in the data dictionary is used. (If you want to specify a format, use the FORMATS= parameter.)

The variables for which you supply formats can be the ones in the *classname*, *variable-label-pairs*, BY=, GROUP=, LABELS= SUBGROUP=, and WEIGHT= parameters.

HTMLDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see “Global and Local Macro Variables” on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

HTMLURL=*URL-specification*

specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **welcome.htm**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**welcome.htm** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **HTMLURL=http://www.reporter.com/reports** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhurl=http://www.reporter.com/reports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name* | *PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT,

then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*
specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myiurl=http://www.reporter.com/reports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **`&`** obtains the value of the macro variable, and the **`%str()`** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LABELS=(*var-label-pairs*)
specifies the paired list of variables that are used in this report definition and the labels to display for these variables on the report output. You must enclose the list in parentheses. Enclose the label in matched single or double quotation marks. If the body of the label contains an unmatched single quotation mark, then use matched single quotation marks to enclose the label and use two single quotation marks to indicate the unmatched single quotation mark or wrap the label in %NRSTR(). For example, both

```
'car''s height'
```

and

```
%nrstr(car's height)
```

display as

```
car's height
```

Do not include commas, equal signs, parentheses, or ampersands in the label. Use one or more spaces between the variable label and the next variable name in the list. Here is an example:

```
labels=(cpubusy="CPU BUSY" loadavg="Load Average")
```

In this example you are specifying labels for two variables (**cpubusy** and **loadavg**) that will be used in your report.

This parameter is not required.

You can use this parameter to specify labels for any variable that is used in this report definition. For example, you can use this parameter to specify the label for the analysis variable, group variable, or subgroup variable. If you use this parameter, then do not specify labels by using any other parameter, such as GRP_LAB=, CL_LAB=, or the label portion of the *variable-label-pairs* parameter. If you specify this parameter, then the labels in the other parameters are ignored. *By default, your report uses the labels that are stored with the variables in the PDB's data dictionary.* If you specify this parameter, it does not change the labels that are stored in the PDB's data dictionary. The labels that you specify by using this parameter are saved only with this report definition.

LARGEDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).

- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

  Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)
- a full path:

  Submit this SAS code:

```
options
 appletloc="full-path-on-this-operating-system";
```

where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

```
options
   appletloc="../path";
```

  Or submit this SAS code on Windows:

```
options
   appletloc="..\path";
```

  where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

  Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

  Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **`View Source`**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*

specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **`hh:mm`**, where **`hh`** is hour (using a 24-hour clock) and **`mm`** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **`00:00`**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose

maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

□ *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **12Apr2003**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

MISS=YES | NO
includes missing values as valid levels for class variables. A heading is displayed that identifies the missing value information that appears in the table. *The default value is YES*.

MISSTEXT='*text*'
supplies text to be printed in table cells that contain missing values. The text can be a maximum length of 200 characters.

NCOLS=*line-size*
is the number of characters that fit across the page for printed reports. This value is not the number of columns that appear in a tabular report.

NROWS=*page-size*
is the number of lines on the page for printed reports. This value is not the number of rows that appear in a tabular report.

ORDER=DATA | FORMATTED | FREQ | INTERNAL
specifies the order in which headings for class variable values are displayed in the table. *The default setting is ORDER=INTERNAL*.

DATA
keeps values of class variables in the order in which they are read from the data set.

FORMATTED
orders the values by their formatted representation.

FREQ
orders the values by descending frequency count so that values which occur more frequently appear first in the table.

INTERNAL
orders values in the same order in which they would be ordered by the SORT procedure.

OUTDESC=*output-description*
if OUTMODE=WEB, specifies a string of text that describes the report group. The string can be a maximum of 40 characters and should not contain double quotation marks. When you display the **welcome.htm** page by using your Web browser, all reports that have the same value of the OUTDESC= parameter and the same value of the OUTLOC= parameter are displayed in the same report group. The value of OUTDESC= is used as the name of the report group. *If OUTMODE=WEB*

- *and you have one or more graph reports on your Web page, then OUTDESC= is optional but, if specified, adds a report grouping functionality to your Web page.*
- *and you have one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is optional, but is helpful if you are using this field for other reports on this Web page.*
- *and you have more than one text report in the folder that is specified by HTMLDIR=, then OUTDESC= is required and its value must be unique within the reports in HTMLDIR=.*

If OUTMODE=LPCAT or OUTMODE=GRAPHCAT, then OUTDESC= specifies a string of text that describes the report. The string can be a maximum of 40 characters and should not contain double quotation marks. The description is stored with the report in the catalog that is specified in the OUTLOC= parameter. *If OUTMODE=LPCAT or OUTMODE=GRAPHCAT*,

- *then OUTDESC= is optional.*

If OUTMODE= has any other value, then the OUTDESC= parameter is ignored.

For more information about when and how to use the OUTDESC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTLOC=*output-location*
for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT:* WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.

- *if OUTMODE=LPCAT*: This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

OUTMODE=*output-format*
specifies the output format for the report, where the output format can be specified as *GWINDOW*, *GRAPHCAT*, *GSF*, *WEB*, *LP*, or *LPCAT*. (If you specified OUTMODE=CATALOG for a macro that was used in a prior version of this software, then the value *CATALOG* is automatically changed to *GRAPHCAT*.)

- *For %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPSPEC: GWINDOW is the default value; LPCAT and LP are invalid values.*
- *For %CPPRINT and %CPTABRPT: LP is the default value; GWINDOW, GRAPHCAT, and GSF are invalid values.*
- *For %CPRUNRPT: if any of the report definitions are based on %CPPRINT or %CPTABRPT, then LP is the default value; otherwise, GWINDOW is the default value. There are no invalid values, but the value of OUTMODE= should be appropriate for the report definitions that are specified in the call. (Each call to %CPRUNRPT should specify only the report definitions that are appropriate to the value of OUTMODE= that is specified in that call. Thus, if you have more than one type of destination, you might need several calls to %CPRUNRPT, one for each value of OUTMODE=.)*
- *For %CPSRCRPT: GWINDOW is the default value. There are no invalid values, but the value must be appropriate for the report.*
- *For %CPXHTML: WEB is the default value; all other values are invalid. Because OUTMODE=WEB is built into the %CPXHTML macro, the OUTMODE= parameter must not be specified in the %CPXHTML macro.*

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see “Examples of the OUT*= Parameters” on page 560.

OUTNAME=*filename* | *entry-name*
for a catalog entry, specifies the one-level name of the entry; for a file, specifies the UNIX or Windows or UNIX File System filename or the z/OS flat file name, or output specification for the file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic “To Direct a Report to an External File” in “How the OUT*= Parameters Work Together” on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default name for a graph report depends in the following way on the value of OUTMODE=:

- *if OUTMODE=GWINDOW:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GRAPHCAT:* G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=GSF:* if the report definition has a name, *report_definition_name*; otherwise, G000000*n* (for charts) and GPLOT*n* (for plots, 3D graphs, and spectrum plots)
- *if OUTMODE=WEB:* S000000n.gif (for the thumbnail image); L000000*n*.gif (for the enlarged image, if LARGEDEV= is not JAVA and is not ACTIVEX) or L*n*.gif (for the enlarged image, if LARGEDEV= is JAVA or ACTIVEX).

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT:* This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTNAME= parameter is not specified, then if the report definition has a name, the default value of OUTNAME= is *report_definition_name*.htm; otherwise, the default value of OUTNAME= is PRTRPT*n*.htm (for print reports) and TABRPT*n*.htm (for tabular reports).

Note: Because the GUI uses an algorithm to determine what name to assign to the report, when you produce Web reports from the SAS IT Resource Management client GUI, there is no field in the attributes that is the equivalent of the OUTNAME= parameter. For more information about the algorithm, see “Report Name” at the end of the section “Directing a Report to the Web” in the chapter “Reporting: Working with Report Definitions” in the *SAS IT Resource Management User’s Guide*. △

For more information about when and how to use the OUTNAME= parameter and about “the big picture” related to the OUT*= parameters, see “How the OUT*= Parameters Work Together” on page 551.

Also see “Examples of the OUT*= Parameters” on page 560.

PALETTE=*palette-name*

specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **sasuser.palette.win** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

PROCOPT=*string*

where *string* is any text that is permitted on the PROC statement of the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro. This allows expert SAS/GRAPH users to take advantage of options that are not supported by SAS IT Resource Management macros or new options for which SAS IT Resource Management has not yet added support.

For %CPPLOT1, %CPPLOT2, and %CPSPEC, the corresponding procedure in SAS/GRAPH is PROC GPLOT.

For %CPCCHRT and CPCHART, the corresponding procedure in SAS/GRAPH is PROC GCHART.

For %CPG3D, the corresponding procedure in SAS/GRAPH is PROC G3D.

For %CPTABRPT, the corresponding procedure in the SAS System is PROC TABULATE.

For example, you might specify an annotate data set:

```
%CPPLOT1(....,
         PROCOPT=ANNO=WORK.MYANNO,
         ....);
```

The default value is blank (no value).

For more details about what options are valid on the PROC GPLOT, PROC G3D, and PROC GCHART statements, see your SAS/GRAPH documentation. For information about PROC TABULATE, see your SAS System documentation.

PRTMISS=YES | NO
forces all row and column headings to be the same for all pages of the table. *The default value is NO*.

REDLVL=*PDB-level*
specifies which level of the table you are reporting on: detail, day, week, month, year, or other. *The default is detail*.

- *When you use this parameter with macros other than %CPRUNRPT*, then the value of this parameter is used as a libref. Specify REDLVL=OTHER if you want to specify a SAS data set in the *dataset* parameter. The SAS data set should contain a variable named DATETIME, which represents the datetime stamp for each observation. For more information, see the *dataset* parameter.

  Note: When specifying the data set name by using the *libref* format, you must assign a *libref* before this macro is invoked. For more information, see the LIBNAME statement in the *SAS Language Reference* documentation for your current release of SAS. △

- *When you use this parameter with the %CPRUNRPT macro*, the REDLVL= parameter specifies a value that overrides the value of the REDLVL= parameter that was specified in the underlying report definition(s) being invoked.

ROTATE=LANDSCAPE | PORTRAIT
indicates whether the output should be displayed in landscape or portrait mode. This parameter is valid only for tabular reports that have been produced in graphics mode, such as those that are sent as output to the graph window. *The default value is device dependent*.

RTS=*number*
specifies the number of print positions to allot to all of the headings in the row dimension, including spaces that are used to print outlining characters for the row headings.

SEPS=YES | NO
places horizontal lines in order to separate the column titles and body of the table. Specifying SEPS=NO completely removes these lines from the table; it does not print blank lines. *The default value is YES*.

If you are creating Web reports and you specify OUTMODE=WEB, then the SEPS= parameter is ignored.

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small “thumbnail” GIF image of your report. *The default is SMALLDEV=GIF160*

when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

STATFMT=*format*

specifies a list of formats, with one format for each statistic that is requested in the STAT= parameter. Use spaces to separate the items in the list. Do not use commas in the list. If you do not want to specify a format for a specific statistic, then specify a period (.) as a placeholder.

This parameter is not required.

STATLAB=*'label'*

specifies the labels (as many as 16 characters) to print in the row or column headers instead of the name of the statistic. Use blanks to separate multiple labels. If you specify this parameter, then you should specify a label for each statistic that is specified in the STAT= parameter on this macro. Use a period to indicate a missing label; here is an example:

```
stat=min mean max,
statlab=least . most
```

To include embedded blanks in a label, enclose the blanks in single or double quotation marks. For example, to use a label such as Monthly Total, you could specify the following:

```
STAT=min mean sum, STATFMT=F8.2 . F12.0,
STATLAB=Minimum Average "Monthly Total"
```

Or, for the STATLAB= parameter, you could specify the following:

```
STATLAB="Minimum" "Average" "Monthly Total"
```

Do not include commas, equal signs, parentheses, or ampersands in the labels. To get a blank label for a statistic, specify the label as " ".

There is a blank character enclosed in the quotation marks.

STMTOPT=*string*

where *string* is any text that is permitted as an option for the statement in the underlying SAS/GRAPH procedure that is used by the SAS IT Resource Management reporting macro.

For %CPPLOT1 and %CPPLOT2, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPCCHRT and %CPCHART, the statement corresponds to the value of the TYPE parameter. For example, if TYPE=HBAR, then the statement will be the HBAR statement in PROC GCHART, and *string* will be added after the '/' in that statement.

For %CPSPEC, the corresponding statement in SAS/GRAPH is the PLOT statement in PROC GPLOT.

For %CPG3D, the corresponding statement in SAS/GRAPH is the SCATTER statement in PROC G3D.

For %CPTABRPT, the corresponding statement in SAS is the TABLE statement in PROC TABULATE.

If you want to customize the labeling and details of the axis for the group variable, you can specify an axis statement such as AXIS3 and then instruct %CPCHART to use it:

```
AXIS3 LABEL=(COLOR=BLUE);
%CPCHART(table,
         ....
         GROUP=gvar,
         STMTOPT=GAXIS=AXIS3,
         ....);
```

The default value is blank (no value).

For more details about what options are available for these statements, see the information about PROC GCHART, PROC G3D, and PROC GPLOT in your SAS/GRAPH documentation and PROC TABULATE in your SAS System documentation.

SUMMARY=NONE | ALL | LAST
indicates the type of summary statistics to provide for this report. The SUMMARY= parameter is not valid if TYPE=TYPE1. If the value of the TYPE= parameter for this report macro is TYPE2, TYPE3, TYPE4, TYPE5, or TYPE6, then valid values for SUMMARY= are ALL or NONE. SUMMARY=LAST is valid only when TYPE=TYPE5 or TYPE6 and you select multiple class variables for this report.

Valid values are as follows:

NONE
indicates that no summary information is provided for the report. *This is the default.*

ALL
indicates that summary information is provided after each class variable and that a summary total for all class variables is listed at the bottom of the report matrix. Here is an example:

Table 3.2 Report Example for SUMMARY=ALL

	Variables	
	Statistics	
Class (nested)		
Summary		
Class (nested)		
Summary		
Summary Total		

LAST
indicates that summary information is provided for the last class variable. Here is an example:

Table 3.3 Report Example for SUMMARY=LAST

	Variables	
	Statistics	
Class (nested) Class (nested) Summary		

TABTYPE=INTERVAL | EVENT
specifies whether each observation in the data that is read into the table represents a specified interval of time or whether each observation was logged in response to an event, such as when a job began or ended. *If the data is in a view or data set in the detail, day, week, month, or year level of the active PDB, then the default is the value that is specified in the table's definition in the active PDB's data dictionary.*

INTERVAL
each observation represents an interval of time

EVENT
each observation represents an event.

TYPE=TYPE1 | TYPE2 | TYPE3 | TYPE4 | TYPE5 | TYPE6
indicates which type of TABULATE table you want to produce. The types are described here:

TYPE1 Analysis X Statistic:

Rows: one row for each analysis variable (as specified by the var1–var10 parameters).

Columns: one column for each statistic (as specified by the STAT= parameter)

TYPE2 Class(value) X Analysis(Statistic):

Rows: one group of rows for each class variable (as specified by the CL_NAM parameter). Within a class variable's group of rows, there is one row for each unique value of that class variable.

Columns: one group of columns for each analysis variable (as specified by the var1–var10 parameters). Within an analysis variable's group of columns, there is one column for each statistic (as specified by the STAT= parameter).

TYPE3 Class(value) X Statistic(Analysis):

Rows: same as TYPE2.

Columns: One group of columns for each statistic (as specified by the STAT= parameter). Within a statistic's group of columns, there is one column for each analysis variable (as specified by the var1-var10 parameters).

TYPE4 Class(value) X Analysis(Sum):

Rows: same as TYPE2.

Columns: same as TYPE2, but there is only one statistic: sum.

TYPE5 Class values X Class (Analysis(Statistic)):

Rows: one row for each combination of the unique values of the class variables (as specified by the CL_NAM= parameter). The rows are sorted in order of the values of the first class variable, and within that, the values of the second class variable, and, within that, the values of the third class variable, and so on.

Columns: one group of columns for each variable specified in the ACROSS= parameter. Within each across variable's group of columns, there is one subgroup for each analysis variable (as specified by the var1–var10 parameters). Within an analysis variable's subgroup of columns, there is one column per statistic (as specified by the STAT= parameter).

TYPE6 Class values X Analysis(Statistic):

Rows: same as TYPE5.

Columns: one group of columns for each analysis variable (as specified by the var1–var10 parameters). Within an analysis variable's group of columns, there is one column per statistic (as specified by the STAT= parameter).

Using parameters, you can define the size of the printable area, the method for handling missing values, several attributes of the headers, and the types of summary rows that you want to see.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

WHERE=*where-expression*
specifies an expression that is used to subset the observations. This expression is known as the local WHERE expression.

Valid operators include but are not limited to BETWEEN ... AND ..., CONTAINS, LIKE, IN, IS NULL, and IS MISSING. (Note: Do not use the ampersand (&) to mean AND in WHERE expressions.) For example, the following expression limits the included observations to those in which the value of the variable MACHINE is the string *host1* or the string *host2* (case sensitive):

```
where=machine in
('host1','host2')
```

In the following example, the expression limits the included observations to those where the value of the variable MACHINE contains the string *host* (case sensitive):

```
where= machine contains 'host'
```

The global WHERE is set with the global macro variable CPWHERE. By default, the local WHERE expression overrides the global WHERE expression, if any. (This default is equivalent to the default *INSTEAD OF* on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI.) If you want the WHERE expression to use both the local WHERE and the global WHERE, insert the words **SAME AND** in front of the WHERE expression in the local WHERE. (*SAME AND* is equivalent to selecting **AND** on the **Query/Where Clause Builder tab** in a report definition that is created in the client GUI). Here is an example:

```
where=SAME AND machine contains 'host'
```

When the global WHERE expression is related to the local WHERE expression with a SAME AND, the WHERE expressions must be compatible for any data to satisfy both expressions. For example, no report is generated if one WHERE expression has MACHINE="Alpha" and the other WHERE expression has MACHINE="Beta". For more information about WHERE and CPWHERE expressions, refer to "Global and Local Macro Variables" on page 6. See also the WHERE statement in the *SAS Language Reference* documentation for your current release of SAS.

Note: On the %CPRUNRPT macro, if the WHERE= parameter is specified, then the value of that parameter overrides the local WHERE expression on each of the report definitions that %CPRUNRPT runs. △

If no local WHERE expression is specified, then the global WHERE expression is used (if CPWHERE is has a non-null value).

%CPTABRPT Notes

You can specify both a label and a format for each analysis variable that you include in this report.

If you specify OUTMODE=WEB for this macro, then specify the OUTNAME= parameter. If you do not specify the OUTNAME= parameter, then the output name defaults to the report definition name (for a saved report definition) or to TABRPT*n*.htm (for a new report).

You can use this macro to produce the following report styles:

- %CPTABRPT Style 1
  - There is one row for each analysis variable and one column for each statistic.
  - You can specify 1–10 analysis variables. You can specify 1–5 statistics.
- %CPTABRPT Style 2
  - There is a set of rows for each class variable. Within each set of rows, there is one row for each value of that class variable.
  - There is a set of columns for each analysis variable, and within each set of columns there is one column for each statistic.
  - You can specify 1–5 class variables, 1–10 analysis variables, and 1–5 statistics.
- %CPTABRPT Style 3
  - There is a set of rows for each class variable. Within each set of rows, there is one row for each value of that class variable.

- There is a set of columns for each statistic, and within each set of columns there is one column for each analysis variable.
- You can specify 1–5 class variables, 1–10 analysis variables, and 1–5 statistics.

- %CPTABRPT Style 4
  - There is a set of rows for each class variable. Within each set of rows, there is one row for each value of that class variable.
  - There is one column for each analysis variable, and that column is for the SUM statistic.
  - You can specify 1–5 class variables and 1–10 analysis variables.
- %CPTABRPT Style 5
  - There is one row for each unique combination of values of the class variables. The rows are sorted in order of the values of the first class variable and, within that, the values of the second class variable, and so on.
  - There is one set of columns for each across variable and within each set of columns, there is one subset for each analysis variable. Within each subset, there is one column for each statistic.
  - You can specify 1–5 class variables, 1–5 across variables, 1–10 analysis variables, and 1–5 statistics.
- %CPTABRPT Style 6
  - There is one row for each unique combination of values of the class variables. The rows are sorted in order of the values of the first class variable and, within that, the values of the second class variable, and so on.
  - There is one set of columns for each analysis variable, and within each set of columns there is one column for each statistic.
  - You can specify 1–5 class variables, 1–10 analysis variables, and 1–5 statistics.
- For all report styles, you can specify *n* optional BY variables. BY variables are not required. If there is one BY variable, then a separate page of the report is generated for each value of the BY variable. If there is more than one BY variable, then a separate page of the report is generated for each unique combination of the values of the BY variables.

%CPWEBINI

Clears automatically generated files from an OUTMODE=WEB output directory.

%CPWEBINI Overview

When you generate a Web-enabled report, you specify the intermediate location of the report with the OUTLOC= parameter and the final location of the report with the HTMLDIR= parameter (and, optionally, the IMAGEDIR= parameter). This macro enables you to clear reports from those locations.

In one call to this macro, you can clear the catalog and one directory (or PDS). If you specify the IMAGEDIR= parameter when you generate the reports, then you need a second call to the %CPWEBINI macro in order to clear the second directory (or PDS).

%CPWEBINI Syntax

%CPWEBINI(

<**CAT=***libref.catalog*>
<,**DIR=***directory-or-PDS*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEURL=***URL-specification*>
<,**LIST=**YES | NO>
<,**_RC=***macro-var-name*>
<,**WEBSTYLE=***style*>);

Details

CAT=*libref.catalog*
specifies the two-level catalog name of the catalog that you want to clear. (This is the SAS catalog that you specified with the OUTLOC= parameter on the report definition(s) that generated the reports that you want to clear.)

The libref must be defined by the time that the macro is called.

DIR=*directory-or-PDS*
specifies the full path and name of a directory or the fully qualified name of a PDS that was specified as either the HTMLDIR= or IMAGEDIR= location on the original report definition. If you want to clear both the HTMLDIR= location and the IMAGEDIR= location, then you must call this macro two separate times, one for each location.

Note: The static files are not deleted when %CPWEBINI clears the reports. For more information about the static files, see the topic "Customizing a Report Gallery's Static Files" in the chapter "Reporting: Working with Galleries" in the *SAS IT Resource Management User's Guide*. △

Note: If the location that is specified by DIR= exists but is empty, a gallery is created and initialized (with control files but not reports) in that location. △

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **`welcome.htm`**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**`welcome.htm`** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **`HTMLURL=http://www.reporter.com/reports`** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **`&`** obtains the value of the macro variable, and the **`%str()`** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.

- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **`&`** obtains the value of the macro variable, and the **`%str()`** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LIST=YES | NO
specifies whether or not to print in the SAS log the list of the files that were deleted. *The default is YES.*

If you specify YES, then the list of files that were deleted is printed in the SAS log. If you specify NO, then this list is not printed in the SAS log.

_RC=*macro-var-name*
specifies the name of a macro variable that is to contain the return code from this macro. The name must start with a letter, can include letters and numbers, and can be eight characters long. To prevent conflicts with the names of SAS IT Resource Management macros, avoid creating variables with names that begin with the character pair CP, CM, CS, CV, or CW (unless specifically shown in SAS IT Resource Management documentation).

Note: Do not use _RC as the name of the macro variable. △

If the macro variable that you specify already exists, then its value will be overwritten. If the macro variable does not exist, then it will be created and will be given a value.

For example, you can specify the variable name RETCODE to store the return code value from this macro. A value of zero indicates a successful execution of the macro. A nonzero value indicates that the macro failed; in this case you can check the explanatory message in the SAS log for more information.

You might want to test this value by using the %CPFAILIF macro (in true batch mode), the %CPDEACT macro (in the SAS Program Editor window), or the %CPLOGRC macro (in true batch mode).

There is no default value for the name of the macro variable.

WEBSTYLE=*style*
indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter "Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

%CPWEBINI Notes

You can clear reports from the Web catalog and directory in the following ways:

- by using the %CPWEBINI macro
- by setting the WEBCLR parameter to YES when you generate additional reports. You can set the WEBCLR= parameter by using the %CPRUNRPT or %CPXHTML macro.
- by using the %CPMANRPT macro.

Both WEBCLR= and %CPWEBINI clear *all* the reports in the specified locations. To clear reports selectively, use the %CPMANRPT macro.

Note: The %CPWEBINI macro clears a SAS catalog or directory or both. If the image directory is separate from the HTML directory, invoke %CPWEBINI a second time for the second directory and omit the CAT= parameter.

Typically, you clear the previous results from the catalog and directory before you put in the current results.

The CAT= parameter points to the intermediate catalog that is used for preparing Web-enabled reports.

The DIR= parameter points to the directory in which the Web-enabled reports reside.

The static files are not deleted when %CPWEBINI clears the reports. For more information about the static files, see the topic "Customizing a Report Gallery's Static Files" in the chapter "Reporting: Working with Galleries" in the *SAS IT Resource Management User's Guide*. △

If the location that is specified by DIR= exists but is empty, the WEBSTYLE= parameter specifies the style of the gallery that %CPWEBINI builds for the first time. If the location that is specified by DIR= exists and is not empty, the WEBSTYLE= parameter specifies the style of the gallery that %CPWEBINI rebuilds after it deletes the reports.

%CPWEBINI Example

This example clears files from an OUTMODE=WEB output directory that is found in **`f:\webtest`** and clears the intermediate report files from the SAS catalog admin.reports.

```
%cpwebini( dir=f:\webtest,
           cat=admin.reports );
```

%CPXHTML

Generates Web pages, based on the results from the %CPEXCEPT macro

%CPXHTML Overview

The %CPXHTML macro generates Web pages (in HTML files that might have links to image files), based on results that are obtained by the %CPEXCEPT macro. (The %CPEXCEPT macro runs *before* the %CPXHTML macro.)

The %CPEXCEPT macro has a RESULTS= parameter, which specifies the location of the folder where it writes the exception information. The %CPXHTML macro also has a RESULTS= parameter, which specifies the location of that same folder. The %CPXHTML macro reads the exception information from that folder and calls the %CPRUNRPT macro to generate one or more Web pages.

You can generate any of the following types of output:

- follow-up reports that are specified in the exception rules (by using the RUN_REPORT= command). These reports are displayed on the welcome.htm Web page.
- a default report for each rule that does not specify any follow-up reports. (For more information about default reports, see "%CPXHTML Notes.") This report is displayed on the welcome.htm Web page.
- a report on all exceptions that are found and the rules to which they correspond. This report is displayed on the excwelco.htm Web page.
- a report on all exception rules, with counts of their exceptions and with the ability to drill down on those exceptions. This report is displayed on the allwelco.htm Web page.
- a report on all service-level-related exception rules, with counts of their exceptions and with the ability to drill down on those exceptions. This report is displayed on the srvwelco.htm Web page.

Note: Along with the four main files (welcome.htm, excwelco.htm, allwelco.htm, and/or srvwelco.htm), the %CPXHTML macro also generates navigation files that the browser displays automatically as you make selections in the left and right frames. Avoid displaying the navigation files directly; you might confuse your browser. △

You can use the GENERATE= parameter to specify the type(s) of reports that you want to produce.

%CPXHTML Syntax

%CPXHTML(
HTMLDIR=*directory-name* | *PDS-name*
,**RULEDS=***libref.name*
<,**DEFRPT=**YES | NO>
<,**GENERATE=***output-type*>
<,**HTMLURL=***URL-specification*>
<,**IMAGEDIR=***directory-name* | *PDS-name*>
<,**IMAGEURL=***URL-specification*>
<,**LARGEDEV=***device-driver*>
<,**LTCUTOFF=***time-of-cutoff*>
<,**MAXREPTS=***number*>
<,**OUTLOC=***output-location*>
<,**PALETTE=***palette-name*>
<,**RESULTS=***exception-result-folder*>
<,**SMALLDEV=***device-driver*>
<,**WEBCLR=**YES | NO>
<,**WEBSTYLE=***style*>);

Details

HTMLDIR=*directory-name* | *PDS-name*

specifies the full path and name of the directory or the fully qualified name of the PDS in which to store the HTML files (*welcome.htm* and its associated HTML files, and the .htm file that are produced for a graph report if LARGEDEV=JAVA or LARGEDEV=ACTIVEX, and the HTML file that is produced for a text report). For example, on Windows you might specify HTMLDIR=*c:\www\hreports* to store your HTML files in the *\www\hreports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhdir and have its value be *c:\www\hreports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

  ```
  %let myhdir=c:\www\hreports ;
  ```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLDIR= %str(&myhdir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhdir)* with the value *c:\www\hreports*.

△

To create Web output, this parameter is required.
This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and should have an LRECL (logical record length) of about 260 if the image files (GIF files) are directed by the IMAGEDIR= parameter to a separate directory. If the GIF files are going to the same directory as the HTML files, then a typical value for LRECL is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, then you might prefer to direct the report to a directory in the z/OS UNIX File System instead of to a PDS. For more information, see the PAGEBY= parameter for "%CPPRINT" on page 414 or the BY= parameter for "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **welcome.htm** file in this directory or in the directory to which you FTP the contents of this PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the HTMLDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. Also see "Examples of the OUT*= Parameters" on page 560.

RULEDS=*libref.name*
specifies the location of the folder that contains the exception rule definitions, which will be used (by %CPEXCEPT or by %CPXHTML) to find the exceptions that are recorded in the RESULTS= file.

Note: The %CPEXCEPT macro runs *before* the %CPXHTML macro runs. △

The folder is implemented as a SAS data set. *This parameter is required for the %CPEXCEPT macro. It is also required for the %CPXHTML macro unless the GENERATE= parameter is set only to FOLLOWUP.* (The default for the GENERATE= parameter is ALL, but you can specify a single type or any combination of types.)

Identify the data set by using the two-level format *libref.name*, where *libref* is already defined and *name* is the name of your exception rule data set.

If the libref is not already defined in your SAS session, then you must define the libref (by using a SAS LIBNAME statement) before you run this macro. You must have at least read access to the library that is referenced by the libref.

Note: %CPXHTML determines which exception rules are to be reported as "service level" rules by checking the third through fifth characters of the rulename for "SLA". If these characters are "SLA", then the rule is treated as a service-level rule and it appears in the service-level report that is produced by the GENERATE=SERVICE (or GENERATE=ALL) setting in %CPXHTML.

In addition, the RULEDS= parameter assigns a title to the exception report that is produced by the GENERATE=SUMMARY (or GENERATE=ALL) setting in %CPXHTML. The title of the exception report is the description that was associated with the RULEDS= data set when the data set was created. △

DEFRPT=YES | NO
specifies whether or not a default report is generated for each exception that does not specify a follow-up report with the RUN_REPORT= "command" in the Recommendation field. For more information about the RUN_REPORT= "command", see the Notes section of "%CPXHTML" on page 536.

The default is YES.

GENERATE=*output-type*
specifies which type of Web output is to be generated. Valid values of *output-type* are

Table 3.4 Values of GENERATE= Parameter

Value of GENERATE= Parameter	Output Location
ALL	allwelco.htm, srvwelco.htm, excwelco.htm, welcome.htm

Value of GENERATE= Parameter	Output Location
FOLLOWUP	welcome.htm
SERVICE	srvwelco.htm, excwelco.htm
SUMMARY	allwelco.htm

The default value is GENERATE=ALL.

Note: You can specify multiple values separated by one or more blanks; here is an example:

```
%CPXHTML(GENERATE=SUMMARY FOLLOWUP,
    HTMLDIR=d:\temp\);
```

△

The following describes the Web pages that are generated by these values.

Allwelco.htm

This Web page is created in the directory that is specified by the HTMLDIR= parameter.

The top half of the page represents all the rules whose names have SLA in bytes 3-5. The top half of the page is a duplicate of the srvwelco.htm page.

The bottom half of the page represents all the rules whose names do not have SLA in bytes 3-5. They are formatted in the same way as the SLA rules (columns represent PDB levels and rows represent descriptions and tables).

Note: The bottom half of the page is not a duplicate of the excwelco.htm page. The excwelco.htm page lists only the rules that have exceptions and the allwelco.htm page lists the rules whether or not they have exceptions. △

In both halves of the page, the cells have the number of exceptions (or OK if there are no exceptions). You can drill down on a cell to get the rule's properties. If there are exceptions, you can also get the follow-up reports.

Excwelco.htm

This Web page is created in the directory specified by the HTMLDIR= parameter.

For each rule that runs against a view in the active PDB, if any exceptions are found and if the rule's recommendation ends with the RUN_REPORT= "command"

```
RUN_REPORT=xxx yyy zzz
<, FOLDER=folder_name>
<, BEGIN=SAS_date_time>
<, END=SAS_date_time>
```

where *xxx*, *yyy*, and *zzz* are the names of one or more report definitions in the specified report definition folder, then %CPXHTML runs the report definitions (*xxx*, *yyy*, and *zzz*) and displays the rule's description, message, and number of exceptions in the list on this page. If follow-up report definitions are not specified in the rule definition, a default follow-up report is generated. (For more information about the RUN_REPORT= "command", see the Notes section of "%CPXHTML" on page 536.)

You can drill down on each rule to see all of its properties and to see the follow-up report(s), if any, which are associated with this rule.

Rules without exceptions are not displayed on this Web page.

Srvwelco.htm
This Web page is created in the directory specified by the HTMLDIR= parameter.

If any rules in the folder have names with SLAA in bytes 3-6, an Availability table is displayed on this page. Similarly, if any rules in the folder have names with SLAR in bytes 3-6, a Responsiveness table is displayed on this page. If any rules in the folder have names with SLAT in bytes 3-6, a Throughput table is displayed on this page.

Each of these three tables is formatted in the same way.

Columns
The first column contains a rule description. Typically, the folder has five almost-identical rules with that same description, one rule for each PDB level of the table. Each remaining column represents a different level (detail, day, week, month, and year). If there is a sixth column, it represents the level Other.

Rows
There is a row for each value of the **`Description`** field. If the same description is used for rules on more than one table, each table has its own row.

Each cell of the table has the number of exceptions (or OK if the rule had no exceptions). You can drill down on a cell to get to the rule's properties. If there are exceptions, you can also drill down to the follow-up reports.

Welcome.htm
This Web page is created in the directory specified by the HTMLDIR= parameter.

For each rule that runs against a view in the active PDB, if any exceptions are found and if the rule's recommendation ends with

```
RUN_REPORT=xxx yyy zzz
<, FOLDER=level_name.folder_name>
<, BEGIN=SAS_date_time>
<, END=SAS_date_time>
```

where *xxx*, *yyy*, and *zzz* are the names of one or more report definitions in the specified report definition folder, then %CPXHTML runs the report definitions (*xxx*, *yyy*, and *zzz*) and displays their reports (called *follow-up reports* on this Web page. If follow-up report definitions are not specified in the rule definition, a default follow-up report is generated. (For more information about the RUN_REPORT= "command", see "%CPXHTML" on page 536.)

There is one report group for each value of *Description*. (More than one rule can have the same value for the *Description* field.)

For each report group, there is one report for each rule that has at least one exception. (Rules without exceptions are not displayed on this Web page.)

HTMLURL=*URL-specification*
specifies the location (URL address) for your browser to use when opening the HTML files for your report. You can use a relative URL (relative to the location of **`welcome.htm`**) or an absolute URL. *This parameter is not required.*

The value that you specify is used as a prefix for all references to HTML files (**`welcome.htm`** and its associated .htm files). For example, if your reports are stored in the directory *www\reports* on your C: drive (that is, HTMLDIR=*c:\www\reports*) on the Windows server that is named *www.reporter.com*, then you might specify **`HTMLURL=http://www.reporter.com/reports`** as the URL for the HTML files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myhurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myhurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *HTMLURL= %str(&myhurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myhurl)* with the value *http://www.reporter.com/reports*.

△

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files (to things such as images) are relative to the directory in which the HTML files reside. In other words, when a URL is not coded in your HTML file, the HTML file expects to find any linked files or images in the same directory where that HTML file is stored. When you store all your images and HTML files in one location, you do not need to specify the HTML URL and you can easily move the entire directory without breaking links.

If you specify this parameter, then the URL is hard-coded in your HTML source. You can use this parameter when your HTML files and image files are not stored in the same location. When this is the case, you must be careful when moving the files so that you do not break any of the links. The HTML file will look for the linked files *only* in the location that is specified in this URL.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEDIR=*directory-name | PDS-name*
specifies the full path and name of the directory or the fully qualified name of the PDS in which to store one or two GIF files for your report (if your report is a graph report). If welcome.htm and its associated HTML files use icons, then the GIF files for the icons are also stored here. For example, on Windows you might specify IMAGEDIR=*c:\www\greports* to store your GIF files in the *\www\greports* directory on your C: drive.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myidir and have its value be *c:\www\greports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myidir=c:\www\greports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEDIR= %str(&myidir)* in the call to any report macro whose report is (or reports are) to go to this location. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myidir)* with the value *c:\www\greports*.

△

This parameter is not required. If you do not specify this parameter, then the value of the HTMLDIR= parameter is used by default. Typically, IMAGEDIR= is not specified.

This parameter is sensitive to environment.

- On UNIX and Windows: The directory or folder must already exist and you must have write access to it.
- On z/OS, if you direct the reports to a z/OS UNIX File System: The directory must already exist and you must have write access to it.

  Note: If you want to send reports directly to z/OS UNIX File System files, then after you call the %CPSTART macro and before you call the report macro you must specify OSYS as the global macro variable CPOPSYS. For more information about CPOPSYS, see "Global and Local Macro Variables" on page 6. △
- On z/OS, if you direct the reports to a PDS (and then FTP the reports to a directory or folder by calling the %CMFTPSND macro): The PDS must already exist and you must have write access to it.

  This PDS should be RECFM=VB (variable blocked) and a typical value of LRECL (logical record length) is 4096. You might need to increase this value depending on the complexity of the individual graphs.

  Note: If you use OUTMODE=WEB and you use either the BY= parameter in a call to %CPPRINT or the PAGEBY= parameter in a call to %CPTABRPT, then you should direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because the report name can be too long to be used as a member name in the PDS. For more information, see the BY= parameter on "%CPPRINT" on page 414 or the PAGEBY= parameter on "%CPTABRPT" on page 507. △

When you want to browse a report that is stored in this location, you can start by pointing your Web browser to the **`welcome.htm`** file in the corresponding HTMLDIR= directory or in the directory to which you FTP the contents of the HTMLDIR= PDS (by using the %CMFTPSND macro).

If IMAGEDIR= and HTMLDIR= point to the same location, then you do not need to specify the HTMLURL= parameter and you do not need to specify the IMAGEURL= parameter. Sharing this location is convenient, and also enables you to easily move the directory as needed.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML. For more information about when and how to use the IMAGEDIR= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

IMAGEURL=*URL-specification*

specifies the location (URL address) that is used in your HTML output to locate your report images. In **`welcome.htm`** and its associated HTML files, the value that you specify is used as a prefix for all references to GIF files. You can specify a relative URL (relative to the location of welcome.htm) or an absolute URL.

This parameter is required if you do not specify the same value or location for HTMLDIR= and IMAGEDIR=. If you do not specify this parameter, the HTML files look for the images in the directory where your HTML output is stored. If the value of IMAGEDIR= and the value of HTMLDIR= are different, the HTML files cannot display the images unless you provide the location of the images by specifying IMAGEURL=.

A URL is an Internet Web address that identifies where a file is located. If you do not specify this parameter, then the links or file addresses in your HTML files

(that link to images) are relative to the directory in which the HTML files are placed. This parameter enables you to identify a specific location or address where the images are stored.

For example, if your report images are stored in the directory *www\reports* on your C: drive (that is, IMAGEDIR=*C:\www\reports*) on the Windows server named *www.reporter.com*, then you might specify *IMAGEURL=http://www.reporter.com/reports* as the URL for the GIF files, if *WWW* is the "root" for the server.

Note: If you prefer to use a macro variable for the value, you can. For example, suppose that you want to make a macro variable named myiurl and have its value be *http://www.reporter.com/reports*.

- In the batch job for reporting, after the call to %CPSTART and before the call to any report macro that will refer to the macro variable, add this code:

```
%let myiurl=http://www.reporter.com/reports ;
```

  This code creates the macro variable, names it, and assigns a value to it.
- Specify *IMAGEURL= %str(&myiurl)* in the call to any report macro whose report is (or reports are) to use this URL. The **&** obtains the value of the macro variable, and the **%str()** is required so that the macro variable is evaluated at the appropriate time during the report production.
- When the job executes and generates the reports, the macro processor replaces *%str(&myiurl)* with the value *http://www.reporter.com/reports*.

△

If the value of IMAGEDIR= is the same as the value of HTMLDIR= (that is, if you store all report files in the same location), then you can easily move all files to a new location without breaking any of the "links" that are coded in the HTML files. If you store your HTML files and IMAGE files in separate directories or PDSs, then your links from the HTML to the image files might not work if you move the files.

This parameter is valid only if you specify OUTMODE=WEB or if the macro is %CPXHTML.

For more information about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551.

LARGEDEV=*device-driver*

specifies the name of the SAS/GRAPH device driver to use in order to create

- an enlarged GIF image of the report, if the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX*. When users click on the thumbnail report in their Web browsers, they see a larger version of the graph report. The larger version is static.

  The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), GIF733 (733x550), and IMGJPEG (JPEG/JFIF 256-color image).
- an ActiveX control, if LARGEDEV=ACTIVEX. When users click on the thumbnail report in the Web browsers, the graph report is displayed by using an ActiveX control. The ActiveX control provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the shortcut menu by clicking the right mouse button.)
- a Java applet, if LARGEDEV=JAVA. When users click on the thumbnail report in their Web browsers, the "life-size" report is displayed by using a Java applet. The applet provides some ability to dynamically manipulate the graph, including drill-down capability if the report definition includes subgroups. (For additional customization options, the user can access the

shortcut menu by clicking the right mouse button.) For more information about using LARGEDEV=JAVA, see the note below.

The default is LARGEDEV=GIF733.

The LARGEDEV= parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the LARGEDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

Note: If a report is generated from a report definition that has LARGEDEV=JAVA, then the mechanism for displaying the report in a Web browser requires read access to a Java archive file named graphapp.jar. On your system, the path to this file is available from the SAS system option APPLETLOC. When you generate the report, your system's value for APPLETLOC is stored with the report (as the value of the variable CODEBASE). When a user views the report through a Web browser, the location is available from CODEBASE. The location must be one that the browser has read access to and that is meaningful to the user's operating system.

To check what your value of APPLETLOC is, submit this SAS code:

```
proc options option=appletloc;
run;
```

This code writes your value of APPLETLOC to your SAS log. (For more information about submitting SAS code, see the section "Working with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.)

If some report users do not have read access to that location, then change the permissions to give them read access, or copy the file to a location that does provide read access. If you copy the file to a different location, then change your value of APPLETLOC to one of the following values:

- a URL:

  Submit this SAS code:

  ```
  options
    appletloc=
        "http://path-to-copy-of-graphapp-jarfile";
  ```

  where *path-to-copy-of-graphapp-jarfile* is the path and name of the directory or folder that contains the file graphapp.jar.

  Check that the path is described in terms that are meaningful to all operating systems. Paths in the form *http:...* are recommended. (For example, if your value of APPLETLOC begins with the characters **C:**, that is not a meaningful address for a user whose Web browser is on a UNIX system.)

- a full path:

  Submit this SAS code:

  ```
  options
   appletloc="full-path-on-this-operating-system";
  ```

  where *full-path-on-this-operating-system* is the full path and name of the directory or folder that contains the file graphapp.jar.

  Using a full path assumes that all of your users are on the same operating system, so that the full path is meaningful for all users. If that assumption is not correct, use a relative path or, even better, use a URL.

- a relative path:

  Submit this SAS code on UNIX:

```
options
   appletloc="../path";
```

Or submit this SAS code on Windows:

```
options
   appletloc="..\path";
```

where *path* is the relative path (with respect to welcome.htm) and name of the directory or folder that contains the file graphapp.jar.

Using a relative path assumes that all of your users are on UNIX and/or Windows operating systems, so that the relative path is meaningful for all users. If that assumption is not correct, use a URL.

Using a relative path offers flexibility. For example, with relative path support, you can zip and move your entire gallery to another location without concern for breaking your Java applets.

To view the value of CODEBASE in a report that was previously created with LARGEDEV=JAVA, double-click on the thumbnail report in your Web browser. The full-size report opens. Right-click near the edge of the report (outside the area of the graph itself). A menu opens. From the menu, select **View Source**. A window opens that displays the source code for the report. In the code, scroll down to the APPLET tag. Within the APPLET tag, the CODEBASE= attribute and its value are on the third line. △

LTCUTOFF=*time-of-cutoff*
specifies a time that the macro uses in order to decide which date is to be used as the value of the keyword LATEST, if the keyword LATEST is used in the specification of the BEGIN= and/or END= parameters. (That is, the LTCUTOFF= parameter is applicable only where the keyword LATEST is used.) The value of LTCUTOFF affects only the date part of the datetime range; it has no effect on the time part of the datetime range.

The time-of-cutoff is specified as a valid SAS time, such as **hh:mm**, where **hh** is hour (using a 24-hour clock) and **mm** is minutes. If the keyword LATEST is used and the LTCUTOFF= parameter is not specified, then the value of LTCUTOFF= defaults to **00:00**.

For more information about specifying the value of the LTCUTOFF= parameter and about how the LTCUTOFF= parameter is used, see the BEGIN= parameter. *The LTCUTOFF= parameter is optional.*

You can use the LTCUTOFF= parameter to determine the value of the LATEST datetime as shown in the following examples. LATEST can be assigned a different date value depending on the time values of the observations in the table, as shown in the two cases that follow this call to the %CPIDTOPN macro.

```
%CPIDTOPN(  ...,BEGIN=LATEST,END=LATEST,LTCUTOFF=4:15);
```

- *Example 1: Latest observation's time is earlier than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:01:30'dt**, then **11Apr2003** is used as the value of LATEST.

  Explanation: Because the time part (01:30) of the latest datetime is not greater than the value of LTCUTOFF= (4:15), SAS IT Resource Management uses the previous date, **11Apr2003**, as the value of LATEST.

- *Example 2: Latest observation's time is later than the value of LTCUTOFF=.* When the report definition runs against a table whose maximum value of DATETIME in the specified level is **'12Apr2003:04:31'dt**, then **12Apr2003** is used as the value of LATEST.

Explanation: Because the time part (4:31) of the latest datetime is greater than the LTCUTOFF value (4:15), SAS IT Resource Management uses the current date, **`12Apr2003`**, as the value of LATEST.

Note: For %CPXHTML:

If the value of LTCUTOFF= is specified in the call to %CPXHTML and the GENERATE= parameter is also specified in the call to %CPXHTML and has the value *ALL* or includes the value *FOLLOWUP*, then %CPXHTML handles each exception in the same way: for every exception, it uses the value of LTCUTOFF= that was specified in the call to %CPXHTML.

If the value of LTCUTOFF= is not specified in the call to %CPXHTML, then %CPXHTML handles each exception differently: for each exception, it uses the value of LTCUTOFF= that was specified in the exception's rule.

(The exceptions are read from the Results data set that was generated by a call to %CPEXCEPT.) △

MAXREPTS=*number*

specifies the maximum number of follow-up reports that are to be generated by this call to the %CPXHTML macro. After the maximum number has been generated, generation of follow-up reports is discontinued but other exception-related HTML generation continues. *The default value of MAXREPTS= is 50.*

To describe this more fully, when you define an exception rule you can optionally specify in its recommendation field the location and name of one or more report definitions that you would like to run if an exception is detected. The reports generated by running these report definitions are called follow-up reports. A counter is set to 0 at the beginning of the call to %CPXHTML. It increments by 1 for each follow-up report for each exception of each rule that has at least one follow-up report.

For example, suppose that exception rule QNAVLBYT has the following line at the end of its recommendation field:

```
run_report=QNMEMUSG QNAVLBYD, folder=admin.itsvrpt
```

Suppose that you run only the QNAVLBYT rule and it detects 10 exceptions. If MAXREPTS=8, then reports will be generated from QNMEMUSG and QNAVLBYD for exceptions 1 of 10 through 8 of 10, but not for exceptions 9 of 10 and 10 of 10.

OUTLOC=*output-location*

for graph reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a graphics stream file (in a format suitable for your operating system); for text reports, specifies the name of a SAS catalog (in the format *libref.catalog*) or specifies the UNIX or Windows or UNIX File System pathname or the z/OS high-level qualifiers or output class name for a text file (in a format suitable for your operating system). For more information about the format suitable for your operating system, see the topic "To Direct a Report to an External File" in "How the OUT*= Parameters Work Together" on page 551.

For graph reports, this parameter is not required. If you do not specify this parameter, then the default location for a graph report depends in the following way on the value of OUTMODE=

- *if OUTMODE=GWINDOW:* WORK.GSEG catalog
- *if OUTMODE=GRAPHCAT*: WORK.GSEG catalog
- *if OUTMODE=GSF:* !SASROOT
- *if OUTMODE=WEB:* WORK.CPWEB_GR catalog.

For text reports, this parameter is omitted in one mode (in order to stay in the intended mode), required in two modes (in order to stay in the intended mode), and optional in one mode.

- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are not specified:* This is the mode in which the report is directed to a SAS window. Omit the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LPCAT:* This is the mode in which the report is directed to a SAS catalog. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=LP, and OUTLOC= and OUTNAME= are specified:* This is the mode in which the report is directed to an external file. Specify the OUTLOC= and OUTNAME= parameters.
- *if OUTMODE=WEB:* This is the mode in which the report is directed to the WEB. Optionally, specify the OUTLOC= and OUTNAME= parameters. If the OUTLOC= parameter is not specified, the metadata about the report goes to the WORK.CPWEB_GR data set.

Note: WORK is a temporary SAS library that exists during your current SAS session. If you want to age out reports from a catalog (by using the %CPMANRPT macro), the libref that is in the value of OUTLOC= must point to a permanent library. For example, you can use the libref ADMIN (which points to the active PDB's ADMIN library) or the libref SASUSER (which points to your SASUSER library), or you can use the SAS LIBNAME statement to create a libref that points to some other permanent SAS library. For more information about the LIBNAME statement, see the documentation for your version of SAS. △

Note: !SASROOT is the location where the SAS software is installed on your local host. △

For more information about when and how to use the OUTLOC= parameter and about "the big picture" related to the OUT*= parameters, see "How the OUT*= Parameters Work Together" on page 551.

Also see "Examples of the OUT*= Parameters" on page 560.

PALETTE=*palette-name*

specifies the name of the palette to use for this report definition. You can specify the name as a three-part name in the format *libref.catalog.entryname*, or you can specify only the entry name of the palette. For example, you can specify **`sasuser.palette.win`** if the palette is stored in your SASUSER library, in a palette catalog, and the palette name is *win*. Supplied palettes are located in PGMLIB.PALETTE.

If you specify only a one-part entry name, then the macro searches for that palette name in your palette list and the first palette with the specified name is used. The list of palette folders is saved in your SASUSER library. In batch mode, you must either allocate your SASUSER library or specify a three-part palette name. If you have more than one palette with the same name and you store them in separate catalogs, then it is best to specify the three-part palette name in order to ensure that you get the palette that you expect.

Several predefined palettes are supplied with SAS IT Resource Management, including IBM3179 (for z/OS), WIN (for all Windows releases), XCOLOR (for UNIX or XWindows), MONO (for monochrome printers), PASTEL (for color printers), and WEB (for most Web output). To view a list of palettes, select the following path from the Manage Report Definitions window in the SAS IT Resource Management GUI for UNIX and Windows environments: **Locals ▶ Select Palette**.

If you do not specify a palette name, then your default palette is used. For additional information on setting the default palette, see the section "Specifying/Editing/Viewing the Default Palette Definition" in the chapter "Reporting: Working with Palette Definitions" in the *SAS IT Resource Management User's Guide*.

You must set the default palette through the Manage Report Definitions window of the SAS IT Resource Management GUI, but this default palette is used both in the SAS IT Resource Management GUI and in batch. If you do not specify a default palette and you do not specify a palette for a specific report, then the following rules apply:

- If OUTMODE=WEB or the macro is %CPXHTML, then the WEB palette is used, regardless of your operating environment type.
- If OUTMODE= is not set to WEB and the macro is not %CPXHTML, then the default palette for your operating environment is used (XCOLOR on UNIX; WIN on Windows; no palette on z/OS) if your operating environment type can be determined.

Palettes must be created and modified through the Manage Report Definitions window in the SAS IT Resource Management GUI. However, you can access and use them in batch.

Note: If you want to try out several palette definitions in the GUI, submit

```
options source;
```

in the program editor to write the source code to the log as it is executed. The name of each palette that you apply will then be recorded in the log. You can refer to the log to find the palette name that you want to use. △

RESULTS=*exception-result-folder*
specifies the SAS data set name of the folder that is to contain the exceptions that are found by the %CPEXCEPT macro. Identify the data set by using the two-level name *libref.datasetname*, where libref is already defined. You can define the libref by using a SAS libname statement before you run this macro.

The exception Web pages that are produced by %CPXHTML are generated from this folder of exceptions. The initial default folder is SASUSER.XRESULTS. After the initial use, the default folder is the most recent folder that is used for results either by the %CPEXCEPT or %CPXHTML macro or by the exception subsystem in the GUI.

For the %CPEXCEPT macro,

- the results data set can reside in any existing SAS library to which you have write access, and it can have any name that is unique to the library.
- the data set is created automatically if it does not already exist. (The data set is cleared automatically if it already exists and if it contains any exceptions from a previous run.)

SMALLDEV=*device-driver*
specifies the name of the SAS/GRAPH device driver to use in order to create the small "thumbnail" GIF image of your report. *The default is SMALLDEV=GIF160 when WEBSTYLE=GALLERY. The default value is GIF260 when WEBSTYLE=GALLERY2 or WEBSTYLE=DYNAMIC.*

The choices (and their corresponding image sizes in pixels) include GIF160 (160x120), GIF260 (260x195), GIF373 (373x280), GIF570 (570x480), and IMGJPEG (JPEG/JFIF 256-color image).

This parameter is valid only if OUTMODE=WEB or the macro is %CPXHTML. For more information about when and how to use the SMALLDEV= parameter and about "the big picture" related to OUTMODE=WEB and %CPXHTML, see "How the OUT*= Parameters Work Together" on page 551.

WEBCLR=YES | NO
specifies whether or not to clear reports from the SAS catalog that is specified with the OUTLOC= parameter and the Web output directories (or PDSs) that are

specified with the HTMLDIR= parameter and, optionally, with the IMAGEDIR= parameter. (The **info.htm** file and a few other help and information files are not deleted.)

For the %CPRUNRPT macro, the default value of this parameter is NO. For the %CPXHTML macro, the default value of this parameter is YES.

Note:

- For the %CPRUNRPT macro, you can clear out all the reports by specifying WEBCLR= or by invoking the %CPWEBINI macro. Alternatively, you can clear out selected reports by invoking the %CPMANRPT macro.
- For the %CPXHTML macro, you can clear out all the reports by specifying WEBCLR= or by invoking the %CPWEBINI macro. However, you cannot clear out selected reports by invoking the %CPMANRPT macro.

△

If you specify WEBCLR=YES, then any existing reports in the specified catalog and directory are deleted before any new reports are written to the catalog and directory. However, the static files are not deleted when you specify WEBCLR=YES. For more information about the static files, see the topic "Customizing a Report Gallery's Static Files" in the chapter "Reporting: Working with Galleries" in the *SAS IT Resource Management User's Guide*.

Note: For the %CPRUNRPT and %CPXHTML macros, if you specify WEBCLR=YES and OUTMODE=WEB, but no OUTLOC= parameter is specified, then any reports in the OUTLOC= default location will be cleared. △

If you specify WEBCLR=NO, then existing reports in the specified catalog and directory are not deleted; new reports are added to the existing set of reports. You, the user, assume responsibility for clearing reports. For example, you might specify YES once a week, or you might clear the reports only when a Web report macro fails because of an out-of-time condition, an out-of-space condition, or some other problem that is indicated by an error message in the SAS log.

Note: You can also use the %CPWEBINI macro as an alternate way to clear reports from the Web catalog and directory. In both %CPWEBINI and WEBCLR=, if you clear the catalog, then you can clear or not clear the directory; however, if you clear the directory, then you must clear the catalog. (If you want to delete some but not all reports, then use the %CPMANRPT macro instead of WEBCLR= or %CPWEBINI.) △

This parameter is valid only if you specify OUTMODE=WEB or the macro is %CPXHTML.

WEBSTYLE=*style*

indicates the layout for the gallery of Web reports. The gallery's layout (that is, style) affects the type of frames, the location of titles, and the control options.

Valid values are GALLERY, GALLERY2, and DYNAMIC, with several exceptions: for the %CPXHTML macro, only GALLERY2 and DYNAMIC are valid; for the %CPHTREE macro, only GALLERY2 and DYNAMIC are valid; and when any reporting macro is used to generate reports to the directories or PDSs created by %CPHTREE, only GALLERY2 and DYNAMIC are valid. The default value is GALLERY2.

This parameter is valid if you specify OUTMODE=WEB or if the macro is %CPHTREE, %CPMANRPT, %CPWEBINI, or %CPXHTML.

Note: Another way to specify the gallery style is to omit the WEBSTYLE= parameter and instead to specify the global macro variable named CPWSTYLE. For more information about CPWSTYLE, see "Global and Local Macro Variables" on page 6 and the topic "Changing the Style in an Existing Gallery" in the chapter

"Reporting: Work with Galleries" in the *SAS IT Resource Management User's Guide*.

For more information about when and how to use the OUTMODE= parameter and about "the big picture" related to OUTMODE=WEB, see "How the OUT*= Parameters Work Together" on page 551. △

%CPXHTML Notes

The output from %CPXHTML is directed as if %CPXHTML had a parameter OUTMODE= and the parameter was set to WEB. Because OUTMODE=WEB is "built in" to %CPXHTML, do not specify OUTMODE=WEB in your call to %CPXHTML.

In a rule definition, you can specify follow-up reports by using a RUN_REPORT= command. You must use a SAS IT Resource Management GUI to add or modify a rule's RUN_REPORT= command. For more information about the RUN_REPORT= command and follow-up reports, see the section "Creating a Rule Definition" in the chapter "Reporting: Working with Rule Definitions" in the *SAS IT Resource Management User's Guide*.

How the OUT*= Parameters Work Together

The values of the OUTMODE=, OUTDESC=, OUTLOC=, and OUTNAME= parameters combine to determine the type of output and the name of the output. The exact combination depends on

- the destination (mode) of the report
- whether the report is a graph report or a text report
- whether the report is produced in batch mode or in a SAS IT Resource Management GUI. (Typically, the GUI combination is the exact equivalent of the batch combination.)

For more information, from the following list select the information that applies to your report:

- "To Direct a Report to a SAS Window" on page 551
- "To Direct a Report to a SAS Catalog" on page 552
- "To Direct a Report to an External File" on page 553
- "To Direct a Report to the Web" on page 556.

Also, for some examples that use OUTMODE= with OUTNAME=, OUTLOC=, and OUTDESC=, see "Examples of the OUT*= Parameters" on page 560.

Note: Report style Source (and macros %CPSRCRPT and %CPRUNRPT) can produce graph reports and text reports. Consult the following section (graph report or text report) that is appropriate. △

To Direct a Report to a SAS Window

1 To direct a graph report to a SAS window:

   a In batch mode:

In true batch mode, you cannot direct a graph report to a window. However, with SAS IT Resource Management running interactively, you can submit a call to a report macro through the SAS Program Editor window and thus approximate batch mode. For the OUT*= parameters in the call to the report macro, specify OUTMODE=GWINDOW, do not specify OUTDESC=, do not specify OUTLOC=, and do not specify OUTNAME=.

In the temporary catalog named WORK.GSEG, the report is directed to the catalog entry named G000000*n*.GRSEG (if the report macro is %CPCHART or %CPCCHRT) or to the catalog entry named GPLOT*n*.GRSEG (if the report macro is %CPPLOT1, %CPPLOT2, %CPG3D, or %CPSPEC).

The SAS Graph window on the local host automatically displays the catalog entry, regardless of whether the PDB is local or remote and, if remote, regardless of whether the report definition runs locally or remotely.

b From the client GUI:

In the Report Output Options window, specify **SAS Window**. You do not need to set any attributes.

The report is directed to the catalog entry named WORK.GSEG.G000000*n*.GRSEG (if the report style is Charts) or to the catalog entry named WORK.GSEG.GPLOT*n*.GRSEG (if the report style is Plots, 3D Graphs, or Spectrum).

The SAS Graph window on the local host automatically displays the catalog entry, regardless of whether the PDB is local or remote and, if remote, regardless of whether the report definition runs locally or remotely.

2 To direct a text report to a SAS window:

a In batch mode:

In true batch mode, you cannot direct a graph report to a window. However, with SAS IT Resource Management running interactively, you can submit a call to a report macro through the SAS Program Editor window and thus approximate batch mode. For the OUT*= parameters in the call to the report macro, specify OUTMODE=LP, do not specify OUTDESC=, do not specify OUTLOC=, and do not specify OUTNAME=.

The SAS Output window on the local host automatically displays the report, regardless of whether the PDB is local or remote and, if remote, regardless of whether the report definition runs locally or remotely.

b From the client GUI:

In the Report Output Options window, specify **SAS Window**. You do not need to set any attributes.

The SAS Output window on the local host automatically displays the report, regardless of whether the PDB is local or remote and, if remote, regardless of whether the report definition runs locally or remotely.

To Direct a Report to a SAS Catalog

1 To direct a graph report to a SAS catalog:

a In batch mode:

For the OUT*= parameters in the call to the report macro, specify OUTMODE=GRAPHCAT, specify OUTDESC=*description*, specify OUTLOC=*libref.catalog*, and specify OUTNAME=*entry*.

The report is directed to *libref.catalog.entry*.GRSEG and the entry's description is the one that you specified. If the PDB is local, the entry is on the local host. If the PDB is remote, the entry is on the local host if you do a

submit, and the entry is on the remote host if you do a remote submit. The libref must be appropriate for the local or remote host.

The report is not displayed automatically.

b From the client GUI:

In the Report Output Options window, specify **`SAS Catalog`**. In the attributes, specify a value for the **`Entry Description`** field, the **`SAS Catalog:Library`** field and **`Catalog Name`** field, and the **`Entry Name`** field.

The report is directed to *libref.catalog.entry*.GRSEG and the entry's description is the one that you specified. If the PDB is local, the entry is on the local host. If the PDB is remote, the entry is on the local host if you select **Run ▶ Local** and the entry is on the remote host if you select **Run ▶ Remote**.

The report is not displayed automatically.

2 To direct a text report to a SAS catalog:

a In batch mode:

For the OUT*= parameters in the call to the report macro, specify OUTMODE=LPCAT, specify OUTDESC=*description*, specify OUTLOC=*libref.catalog*, and specify OUTNAME=*entry*.

The report is directed to *libref.catalog.entry*.OUTPUT and the entry's description is the one that you specified. If the PDB is local, the entry is on the local host. If the PDB is remote, the entry is on the local host if you do a submit, and the entry is on the remote host if you do a remote submit. The libref must be appropriate for the local or remote host.

The report is not displayed automatically.

b From the client GUI:

In the Report Output Options window, specify **`SAS Catalog`**. In the attributes, specify a value for the **`Entry Description`** field, the **`SAS Catalog:Library`** field and **`Catalog Name`** field, and the **`Entry Name`** field.

The report is directed to *libref.catalog.entry*.OUTPUT and the entry's description is the one that you specified. If the PDB is local, the entry is on the local host. If the PDB is remote, the entry is on the local host if you select **Run ▶ Local** and the entry is on the remote host if you select **Run ▶ Remote**.

The report is not displayed automatically.

To Direct a Report to an External File

1 To direct a graph report to an external file:

a In batch mode:

For the OUT*= parameters in the call to the report macro, specify OUTMODE=GSF, and do not specify OUTDESC=. Specify OUTLOC= and OUTNAME=. The values for OUTLOC= and OUTNAME= depend on your operating environment.

- For UNIX or Windows: Specify OUTLOC=*pathname* and specify OUTNAME=*filename* or *filename.extension*, where the extension might be *.gsf*.

  The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.

- For z/OS if you use the UNIX File System: Specify OUTLOC=*pathname* and specify OUTNAME=*filename* or *filename.extension*, where the extension might be *.gsf*.

The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.

- For z/OS if you do not use the UNIX File System and if you want to save the report: Specify OUTLOC=*high-level-qualifiers* and specify OUTNAME=*flat_file_name*.

  The report is directed to *high-level-qualifiers*||*flat_file_name*.

- For z/OS if you do not use the UNIX File System and if you want to direct the report to an output device, such as a plotter: Specify OUTLOC=*class_specification*, where *class_specification* might be something like *sysout=b*. Also specify OUTNAME=*device_specification*, where *device_specification* might be something like *dest=c242r1*.

  The report is directed to the specified class and device.

If the PDB is local, the report is on the local host. If the PDB is remote, the report is on the local host if you do a submit, and the report is on the remote host if you do a remote submit. The pathname or high-level qualifiers or class and device specifications must be appropriate for the local or remote host.

Note: In the concatenations, if you want punctuation between the value of OUTLOC= and the value of OUTNAME=, you must provide the punctuation (typically at the end of the value of OUTLOC=). For example, you must provide the period between the high-level qualifiers and data set name in the full name of a z/OS data set, and you must provide the forward slash (/) or backward slash (\) between the pathname and filename in the full name of a UNIX or Windows file. △

b From the client GUI:

In the Report Output Options window, specify **`External File`**. In the attributes, specify a value in the **`File Location`** field and a value in the **`File Name`** field. The values for **`File Location`** and **`File Name`** depend on your operating environment.

- For UNIX or Windows: In the **`File Location`** field, specify a pathname. In the **`File Name`** field, specify a *filename* or *filename.extension*, where the extension might be *.gsf*.

  The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.

- For z/OS if you use the UNIX File System: In the **`File Location`** field, specify a pathname. In the **`File Name`** field, specify a *filename* or *filename.extension*, where the extension might be *.txt*.

  The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.

- For z/OS if you do not use the UNIX File System and if you want to save the report: In the **`File Location`** field, specify *high-level-qualifiers*. In the **`File Name`** field, specify *flat_file_name*.

  The report is directed to *high-level-qualifiers*||*flat_file_name*.

- For z/OS if you do not use the UNIX File System and if you want to direct the report to an output device, such as a plotter: In the **`File Location`** field, specify OUTLOC=*class_specification*, where *class_specification* might be something like *sysout=b*. In the **`File Name`** field, specify *device_specification*, where *device_specification* might be something like *dest=c242r1*.

  The report is directed to the specified class and device.

If the PDB is local, the report is on the local host. If the PDB is remote, the report is on the local host if you select **Run ▶ Local** and the report is on the remote host if you select **Run ▶ Remote**.

Note: In the concatenations, if you want punctuation between the value of the **`File Location`** field and the value of the **`File Name`** field, you must provide the punctuation (typically at the end of the **`File Location`** field). For example, you must provide the period between the high-level qualifiers and data set name in the full name of a z/OS data set, and you must provide the forward slash (/) or backward slash (\) between the pathname and filename in the full name of a UNIX or Windows file. △

2 To direct a text report to an external file:

a In batch mode:

For the OUT*= parameters in the call to the report macro, specify OUTMODE=LP, and do not specify OUTDESC=. Specify OUTLOC= and OUTNAME=. The values for OUTLOC= and OUTNAME= depend on environment.

- For UNIX or Windows: Specify OUTLOC=*pathname* and specify OUTNAME=*filename* or *filename.extension*, where the extension might be *.txt*.

  The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.
- For z/OS if you use the UNIX File System: Specify OUTLOC=*pathname* and specify OUTNAME=*filename* or *filename.extension*, where the extension might be *.txt*.

  The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.
- For z/OS if you do not use the UNIX File System and if you want to save the report: Specify OUTLOC=*high-level-qualifiers* and specify OUTNAME=*flat_file_name*.

  The report is directed to *high-level-qualifiers*||*flat_file_name*.
- For z/OS if you do not use the UNIX File System and if you want to direct the report to an output device, such as a printer: Specify OUTLOC=*class_specification*, where *class_specification* might be something like *sysout=b*. Also specify OUTNAME=*device_specification*, where *device_specification* might be something like *dest=c242r2*.

  The report is directed to the specified class and device.

If the PDB is local, the report is on the local host. If the PDB is remote, the report is on the local host if you do a submit, and the report is on the remote host if you do a remote submit. The pathname or high-level qualifiers or class and device specifications must be appropriate for the local or remote host.

Note: In the concatenations, if you want punctuation between the value of OUTLOC= and the value of OUTNAME=, you must provide the punctuation (typically at the end of the value of OUTLOC=). For example, you must provide the period between the high-level qualifiers and data set name in the full name of a z/OS data set, and you must provide the forward slash (/) or backward slash (\) between the pathname and filename in the full name of a UNIX or Windows file. △

b From the client GUI:

In the Report Output Options window, specify **`External File`**. In the attributes, specify a value in the **`File Location`** field and a value in the

File Name field. The values for **File Location** and **File Name** depend on environment.

- For UNIX or Windows: In the **File Location** field, specify a pathname. In the **File Name** field, specify a *filename* or *filename.extension*, where the extension might be *.txt*.

  The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.

- For z/OS if you use the UNIX File System: In the **File Location** field, specify a pathname. In the **File Name** field, specify a *filename* or *filename.extension*, where the extension might be *.txt*.

  The report is directed to *pathname*||*filename* or *pathname*||*filename.extension*.

- For z/OS if you do not use the UNIX File System and if you want to save the report: In the **File Location** field, specify *high-level-qualifiers*. In the **File Name** field, specify *flat_file_name*.

  The report is directed to *high-level-qualifiers*||*flat_file_name*.

- For z/OS if you do not use the UNIX File System and if you want to direct the report to an output device, such as a printer: In the **File Location** field, specify OUTLOC=*class_specification*, where *class_specification* might be something like *sysout=b*. In the **File Name** field, specify *device_specification*, where *device_specification* might be something like *dest=c242r2*.

  The report is directed to the specified class and device.

If the PDB is local, the report is on the local host. If the PDB is remote, the report is on the local host if you select **Run ▶ Local** and the report is on the remote host if you select **Run ▶ Remote**.

Note: In the concatenations, if you want punctuation between the value of the **File Location** field and the value of the **File Name** field, you must provide the punctuation (typically at the end of the **File Location** field). For example, you must provide the period between the high-level qualifiers and data set name in the full name of a z/OS data set, and you must provide the forward slash (/) or backward slash (\) between the pathname and filename in the full name of a UNIX or Windows file. △

To Direct a Report to the Web

1 To direct a graph report to the Web:

   a In batch mode:

   For the OUT*= parameters in the call to the report macro, specify OUTMODE=WEB, specify OUTDESC=*report_group_name* if you want to use report grouping, specify OUTLOC=*libref.catalog*, and do not specify OUTNAME=. For more information about the requirements that are related to report grouping, see the OUTDESC= parameter.

   You must also specify a value for HTMLDIR= and you might want to specify values for other parameters such as LARGEDEV= (whose value is *JAVA*, *ACTIVEX*, or some other value) and WEBSTYLE=. Specifying IMAGEDIR= is optional; if you do not specify IMAGEDIR=, the value of IMAGEDIR= defaults to the value of HTMLDIR=. (Typically, the value of

IMAGEDIR= is *not* specified.) The value of HTMLDIR= (and, if you use IMAGEDIR=, the value of IMAGEDIR=) must be unique to the value of OUTLOC=. That is, you can think of each catalog as having its own location for its HTML files (and its own location for its GIF files, if you separate the GIF files from the HTML files).

First, the report is directed to *libref.catalog*._000000*n*.GRSEG. (Also, the metadata about the report is directed to an observation in *libref.data_set*, where the libref and name of the SAS data set are the same as the libref and name of the SAS catalog that you specified.)

Then, if the value of IMAGEDIR= is the name of a directory or folder, the thumbnail version of the report is directed to file s000000*n*.gif in that directory or folder. If the value of IMAGEDIR= is the name of a PDS, the thumbnail version of the report is directed to member S000000*n* in that PDS.

Also, if the value of LARGEDEV= is *JAVA* or *ACTIVEX* and if the value of HTMLDIR= is the name of a directory or folder, the applet or control version of the report is directed to file L*n*.htm in that directory or folder. If the value of LARGEDEV= is *JAVA* or *ACTIVEX* and if the value of HTMLDIR= is the name of a PDS, the applet or control version of the report is directed to member L*n* in that PDS. If the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX* and if the value of IMAGEDIR= is the name of a directory or folder, the enlarged size version of the report is directed to file L000000*n*.gif in that directory or folder. If the value of LARGEDEV= is not *JAVA* and is not *ACTIVEX* and if the value of IMAGEDIR= is the name of a PDS, the enlarged size version of the report is directed to member L000000*n* in that PDS.

If the PDB is local, the reports are on the local host. If the PDB is remote, the reports are on the local host if you do a submit, and the reports are on the remote host if you do a remote submit. (If OUTMODE=WEB, do not do a remote submit to a z/OS host.) The pathnames or high-level qualifiers must be appropriate for the local or remote host.

b From the client GUI:

In the Report Output Options window, specify **Web**. In the attributes, specify a value for the **Report Group** field if you want to use report grouping, and specify a value for the **SAS Catalog:Libref** field and **Catalog Name** field. For more information about the requirements that are related to report grouping, see the **Report Group** field in the section "Directing a Report to the Web" in the chapter "Reporting: Working with Report Definitions" in the *SAS IT Resource Management User's Guide*.

You must also specify a value for the **HTML Directory** field and you can select the **Use Interactive Graphics When Possible** field (there is an X in the box when the field is selected), select the type of interactive graphics (**JAVA** or **ACTIVEX**), and specify a value in the **Web Style** field. Specifying a value in the **Image Directory** field is optional; if you do not specify a value in the **Image Directory** field, the value of the **Image Directory** field defaults to the value of the **HTML Directory** field. (Typically, the value of Image Directory is *not* specified.) The value of the **HTML Directory** field (and, if you use the **Image Directory** field, the value of the **Image Directory** field) must be unique to the catalog's libref and name. That is, you can think of each catalog as having its own location for its HTML files (and its own location for its GIF files, if you separate the GIF files from the HTML files).

First, the report is directed to *libref.catalog*._000000*n*.GRSEG. (Also, the metadata about the report is directed to an observation in *libref.data_set*, where the libref and name of the SAS data set are the same as the libref and name of the SAS catalog that you specified.)

Then, if the value of the **Image Directory** field is the name of a directory or folder, the thumbnail version of the report is directed to file s000000*n*.gif in that directory or folder. If the value of the **Image Directory** field is the name of a PDS, the thumbnail version of the report is directed to member S000000*n* in that PDS.

Also, if the **Use Interactive Graphics When Possible** field is selected and the value of the **HTML Directory** field is the name of a directory or folder, the Java applet or ActiveX control version of the report is directed to file L*n*.htm in that directory or folder. If the **Use Interactive Graphics When Possible** field is selected and if the value of the **HTML Directory** field is the name of a PDS, the applet or control version of the report is directed to member L*n* in that PDS. If the **User Interactive Graphics When Possible** field is not selected and if the value of the **HTML Directory** field is the name of a directory or folder, the enlarged size version of the report is directed to file L000000*n*.gif in that directory or folder. If the **Use Interactive Graphics When Possible** field is not selected and if the value of the **HTML Directory** field is the name of a PDS, the enlarged size version of the report is directed to member L000000*n* in that PDS.

If the PDB is local, the reports are on the local host. If the PDB is remote, the reports are on the local host if you select **Run ▶ Local** and the reports are on the remote host if you select **Run ▶ Remote**. (If you specify **Web** in the Report Output Options window, do not select **Run ▶ Remote** if the remote host has a z/OS operating environment.)

2 To direct a text report to the Web:

a In batch mode:

For the OUT*= parameters on the call to the report macro, specify OUTMODE=WEB, specify OUTDESC=*report_group_name* if you want to use report grouping or if you must use report grouping, specify OUTLOC=*libref.catalog* and specify OUTNAME=*filename*. (For more information about the requirements that are related to report grouping, see the OUTDESC= parameter.)

You must also specify a value for HTMLDIR= and can specify values for other parameters such as LARGEDEV= (whose value must not be *JAVA* and must not be *ACTIVEX*) and WEBSTYLE=. Specifying IMAGEDIR= is optional; if you do not specify IMAGEDIR=, the value of IMAGEDIR= defaults to the value of HTMLDIR=. (Typically, the value of IMAGEDIR= is *not* specified.) The value of HTMLDIR= (and, if you use IMAGEDIR=, the value of IMAGEDIR=) must be unique to the value of OUTLOC=. That is, you can think of each catalog as having its own location for its HTML files (and its own location for its GIF files, if you separate the GIF files from the HTML files).

First, the metadata about the report is directed to an observation in the SAS data set named *libref.data_set*, where the libref and name of the SAS data set are the same as the libref and name of the SAS catalog that you specified in OUTLOC=. (The report is *not* also directed to that catalog. The report is directed to the file.)

Then, if the value of HTMLDIR= is the name of a directory or folder, the report is directed to a file named *filename*.htm in that directory or folder. If no value was specified for the OUTNAME= parameter, the report is directed to a file named *rptdef.htm* in that directory or folder, where *rptdef* is the name of the report definition. If the value of HTMLDIR= is the name of a PDS, the report is directed to a member named *filename* in that PDS.

Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter on a call to %CPPRINT or the BY= parameter in a call to

%CPTABRPT, you might want to direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because in the PDS you get a single very large file without the typical filtering and in the directory you get multiple small files with the typical filtering. For more information, see the BY= parameter for "%CPPRINT" on page 414 or the PAGEBY= parameter for "%CPTABRPT" on page 507. △

If the PDB is local, the report or reports are on the local host. If the PDB is remote, the report or reports are on the local host if you do a submit, and the report or reports are on the remote host if you do a remote submit. (If OUTMODE=WEB, do not do a remote submit to a z/OS host.) The pathnames or high-level qualifiers must be appropriate for the local or remote host.

b From the client GUI:

In the Report Output Options window, specify **Web**. In the attributes, specify a value for the **Report Group** field if you want to use report grouping or if you must use report grouping, and specify a value for the **SAS Catalog: Libref** field and **Catalog Name** field. You cannot specify a name for the report. For more information about the requirements that are related to report grouping, see the **Report Group** field in the topic "Directing a Report to the Web" in the chapter "Work with Report Definitions" in the *SAS IT Resource Management User's Guide*.

You must also specify a value for the **HTML Directory** field, and you can specify a value for the **Web Style** field. Specifying a value in the **Image Directory** field is optional; if you do not specify a value in the **Image Directory** field, the value of the **Image Directory** field defaults to the value of the **HTML Directory** field. (Typically, the value of Image Directory is *not* specified.) The value of the **HTML Directory** field (and, if you use the **Image Directory** field, the value of the **Image Directory** field) must be unique to the catalog's libref and name. That is, you can think of each catalog as having its own location for its HTML files (and its own location for its GIF files, if you separate the GIF files from the HTML files).

First, the metadata about the report is directed to an observation in the SAS data set named *libref.data_set*, where the libref and name of the SAS data set are the same as the libref and name of the SAS catalog that you specified. (The report is *not* also directed to that catalog. The report is directed to the file.)

Then, if the value of the **HTML Directory** field is the name of a directory or folder and the report definition has been saved, the report is directed to a file named *rptdef*.htm in that directory or folder, where *rptdef* is the name of the report definition. If the value of the **HTML Directory** field is the name of a PDS and the report definition has been saved, the report is directed to the member named *rptdef* in that PDS, where *rptdef* is the name of the report definition. If the value of the **HTML Directory** field is the name of a directory or folder and the report definition has not been saved, the report is directed to a file named TABRPT*n*.htm (if the report is a tabular report) or to a file named PRTRPT*n*.htm (if the report is a print report) in that directory or folder. If the value of the **HTML Directory** field is the name of a PDS and the report definition has not been saved, the report is directed to a member named TABRPT*n* (if the report is a tabular report) or to a member named PTRRPT*n* (if the report is a print report) in that PDS.

Note: If you use OUTMODE=WEB and you use either the PAGEBY= parameter in a call to %CPPRINT or the BY= parameter in a call to %CPTABRPT, you might want to direct the report to a directory in the z/OS UNIX File System instead of to a PDS, because in the PDS you get a single very large file without the typical filtering and in the directory you get

multiple small files with the typical filtering. For more information, see the BY= parameter for "%CPPRINT" on page 414 or the PAGEBY= parameter for "%CPTABRPT" on page 507. △

If the PDB is local, the report or report are on the local host. If the PDB is remote, the report or reports are on the local host if you select **Run ▶ Local** and the report or reports are on the remote host if you select **Run ▶ Remote**. (If you specify **Web** in the Report Output Options window, do not select **Run ▶ Remote** if the remote host has a z/OS operating environment.)

Examples of the OUT*= Parameters

The following examples illustrate the construction of the destination for a report. The additional commas indicate positional parameters that are omitted.

- *OUTMODE=GWINDOW:*

  OUTMODE=GWINDOW, so the destination of the report is the SAS Graph window.

```
%CPCHART(tababc,machine,,pccpuby,
        type=hbar,
        outmode=gwindow);
```

- *OUTMODE=GRAPHCAT*:

  OUTLOC=SASUSER.CPUPGMS. and OUTNAME=RESULTS, so the destination of the report is the SAS catalog entry named SASUSER.CPUPGMS.RESULTS.GRSEG. That entry has the description *All CPUs Percent Busy (Dispatched)*.

```
%CPCHART(tababc,machine,,pccpuby,
        type=hbar,
        outmode=graphcat,
        outdesc=All CPUs Percent Busy (Dispatched),
        outloc=sasuser.cpupgms,
        outname = results);
```

- *OUTMODE=GSF on z/OS, output to a data set that is not in the z/OS UNIX File System:*

  OUTLOC=SASXYZ. and OUTNAME=RESULTS, so the destination of the graphics stream file is the z/OS data set named SASXYZ.RESULTS.

```
%cpchart(tababc,machine,,pccpuby,,
     type=hbar,
     outmode=gsf,
     outloc=sasxyz.,
     outname= results);
```

- *OUTMODE=GSF on z/OS, output to a printer:*

  OUTLOC=SYSOUT=B and OUTNAME=DEST=C269R1, so the destination of the graphics stream file is the printer named C269R1. The print job is in the class that is defined by SYSOUT=B.

```
%cpchart(tababc,machine,,pccpuby,,
     type=hbar,
     outmode=gsf,
     outloc=sysout=b,
     outname=dest=c269r1);
```

- *OUTMODE=GSF on Windows:*

  OUTLOC=c:\temp\ and OUTNAME= results.gsf, so *c:\temp\results.gsf* is the destination of the graphics stream file.

```
%cpchart(tababc,machine,,pccpuby,,
     type=hbar,
     outmode=gsf,
     outloc=c:\temp\,
     outname= results.gsf);
```

- *OUTMODE=WEB on Windows:*

  Because WEBSTYLE= is not specified, the style defaults to *GALLERY2*. Because LARGEDEV= is not specified, the enlarged report defaults to a GIF image with a size of 733x550 pixels. Because IMAGEDIR= is not specified, the report goes to HTMLDIR=, which is *c:\webrpts*. If this is the fifth report to go in c:\webrpts, then the report files are S0000005.gif (for the thumbnail report) and L0000005.gif (for the enlarged report). You can see the report by pointing a Web browser to the *welcome.htm* file in *c:\webrpts* and looking in the report group named *My Charts*.

```
%cpchart(tababc,machine,,pccpuby,,
     type=hbar,
     outmode=web,
     htmldir=c:\webrpts,
     outloc=work.cpweb_gr,
     outdesc=My Charts);
```

CHAPTER

4

Data Dictionary Macros and Control

Using the %CPDDUTL Macro

The following sections describe the %CPDDUTL macro and all control statements that are associated with the macro. They also describe specific syntax guidelines that are related to this macro and statistical information that is related to variables and formula variables.

%CPDDUTL is used to manage the data dictionary in batch mode. By using the associated control statements with %CPDDUTL, you can create and modify tables, variables, formula variables, derived variables, or the entire data dictionary. This section also provides concepts about %CPDDUTL, the master data dictionary, performance databases, and how other macros act on tables and the active PDB.

- %CPDDUTL - creates, modifies or prints tables, variables, formula variables, derived variables, or the entire data dictionary (see "%CPDDUTL" on page 569)
- %CPDDUTL Control Statements (see "%CPDDUTL Macro Control Statements" on page 571)
- %CPDDUTL Macro Concepts (see "%CPDDUTL Macro Concepts" on page 567)
- %CPDDUTL Syntax Guidelines (see "%CPDDUTL Syntax Guidelines" on page 568)
- Variable Interpretation Type Definitions (see "Variable Interpretation Types" on page 697)
- Variable Interpretation Type Summary Table (see "Variable Interpretation Type Summary Table" on page 710).

%CPDDUTL Macro Concepts

The %CPDDUTL macro enables you to manage the entire performance database (PDB). This is accomplished by using %CPDDUTL control statements that act on the master data dictionary and/or the active PDB's data dictionary or that prepare control statements for this purpose. The active PDB is the PDB that has been specified in the most recent %CPSTART macro.

The master data dictionary contains supplied table definitions and variable definitions. A PDB contains a data dictionary and data that you have processed and reduced into one or more levels of the PDB, such as detail, day, week, month, and year.

A PDB can contain one or more tables. A table consists of the table's definition in the data dictionary and the data in the table at each level of the PDB. A table definition includes information such as the collector that is used to collect data, the table's name and type, and the levels into which data for this table should be summarized. Definitions for variables, derived variables, and formula variables are associated with the table.

Some %CPDDUTL control statements and parameters act on the table's data, without changing or deleting the table's definition. For example, the PURGE TABLE statement purges the data in the detail level of the table, but it does not prevent new

data from being processed or reduced into the table at a later time, and it does not remove the table definition. Other %CPDDUTL control statements act on the table's data and the table's definition. For example, the DELETE TABLE statement deletes the data in the table as well as the definition of the table.

Note: You can remove data with the %CPDDUTL macro. If the %CPDDUTL macro cannot remove the specific data that you want to remove, consider using the %CPEDIT macro. For more information, see "%CPEDIT" on page 99. △

Associated with a table definition are regular, derived, and formula variable definitions. These definitions include information for each variable in the table, such as the variable's name, description, and interpretation type; the data type; and the kind of statistics that are to be calculated for the variable at each of the summary levels in the PDB.

The data that is associated with a table is stored in one or more SAS data sets, depending on the level(s) in which the data is stored in the PDB. For each table and level of the PDB, there is a SAS view that provides a logical picture of the data and the SAS data sets that contain the data.

Control statements that act on the PDB's data dictionary, such as VERIFY DICTIONARY, affect all the tables and variables in the data dictionary.

Control statements that can act on one or more tables, such as DELETE TABLES, affect the specified tables and all the variables within the table.

Control statements that act on a variable affect only the specified variable in the active table in the active PDB. If you have not set the active table, then an error results. You can use the SET TABLE control statement in order to set the table in the active PDB that is to be the active table, as shown in the following example:

```
set table
   name=tab1;
update variable
   name=var1 description='survey';
update variable
   name=var2 format=5.3;
set table
   name=tab2;
update variable
   name=var2 label='automobile';
```

This example updates three variables within two different tables. Both tables contain a variable that is named VAR2. The first control statement that refers to variable VAR2 updates the VAR2 variable that is in table TAB1. The second control statement that refers to variable VAR2 updates the VAR2 variable that is in table TAB2.

%CPDDUTL Syntax Guidelines

The control statements that are executed by %CPDDUTL have the syntax

```
FUNCTION OBJECT PARAMETERS;
```

where

- FUNCTION is always required and can be one of the following, as defined for each specific function: ADD, BUILD, COMPARE, CREATE, DELETE, END, GENERATE, INCLUDE, INSTALL, LIST, MAINTAIN, PRINT, PURGE, QUIT, SAVE, SET, STATUS, STOP, SYNCHRONIZE, VERIFY, UPDATE, XREF.

- OBJECT is usually required and can be one of the following: DERIVED, DERIVEDS, DICTIONARY, DEFAULTS, FORMULA, FORMULAS, SOURCE, SYNTAX, TABLE, TABLES, VARIABLE, VARIABLES, or VIEWS, as applicable to the specified function. The following functions do not require an object: END, QUIT, and STOP.
- PARAMETERS
  - are required, optional, or undefined, depending on the control statement
  - are separated by blanks
  - have one of these forms:

```
keyword
keyword [=] value
```

    If the equal sign is missing in a keyword=*value* pair, then the equal sign is assumed. The value for a keyword=*value* pair must not be omitted. You can omit the spaces around the equal sign in a keyword=*value* pair.
  - can contain other parameters and can be enclosed in parentheses, depending on the control statement. Here is an example:

```
day = (agelimit=5 average
count)
```

  - can contain strings in single quotation marks, depending on the control statement. The text of the string can be as long as 200 characters and must not contain embedded single or double quotation marks. *If necessary, you can break a quoted string in the middle of a word and then continue the word in byte 1 of the next line.*
- at least one blank separates function, object, and each parameter
- each control statement ends with a semicolon (;)
- comments can be listed anywhere that blanks are allowed. A comment can span multiple lines and can be used to prevent entire control statements from running, if required. A comment has this format:

```
/* body of comment
*/
```

Control statements are processed in the order in which they are read. If a control statement results in an error and does not have a NOERROR parameter, then none of the control statements that follow are processed, but the syntax of the remaining control statements is verified.

CAUTION:

Do not use tabs in control statements. △

%CPDDUTL

Manages the entire data dictionary in the active PDB

%CPDDUTL Overview

The %CPDDUTL macro enables you to add, delete, and change information in the active PDB's data dictionary. You can also generate %CPDDUTL control statements to perform these tasks as well as to load table and variable definitions into the master data dictionary.

To run this macro, you write control statements and store them in an external file or in an entry in a SAS catalog. When you run %CPDDUTL, the macro reads and acts on the control statements, which perform functions such as modifying or printing table, variable, and formula variable definitions. When the %CPDDUTL macro acts on most control statements, the data dictionary in the active PDB is updated and views are rebuilt for tables that were changed by the %CPDDUTL control statements.

For additional information about %CPDDUTL, see the following documents: "%CPDDUTL Macro Concepts" on page 567, "%CPDDUTL Syntax Guidelines" on page 568, and "%CPDDUTL Macro Control Statements" on page 571.

%CPDDUTL Syntax

%CPDDUTL(
ENTRYNAM=*SAS-catalog-entry*
,**FILENAME=***'physical-filename'*
<**,LIST=**Y | N>);

Details

ENTRYNAM=*SAS-catalog-entry*
is the name of an entry in a SAS catalog. The entry contains the control statements that the %CPDDUTL macro will run. The name must be a four-level entry name with a type of SOURCE, such as *libref.catalog.member.source*, and cannot be enclosed in quotation marks. *You must specify either this parameter or the FILENAME= parameter, but not both.*

FILENAME=*'physical-filename'*
is the physical location and name of an external file, z/OS data set, or PDS member that is to contain the output or input for this control statement. This includes the complete directory path or high-level qualifiers plus the file or member name. *The name must be enclosed in single quotation marks and must not contain any embedded blanks.* You must specify either this parameter or the ENTRYNAM= parameter, but not both.

CAUTION:
Do not use double quotation marks in the FILENAME= parameter. △

LIST=Y | N
indicates whether or not a listing of your control statements and their line numbers is to be produced in the SAS log. A listing can help you locate errors in your statements because error messages refer to the line numbers of the source lines that contain the errors. This parameter is optional.
The default is N, which does not produce a listing.

%CPDDUTL Notes

You must use the %CPSTART macro with ACCESS=WRITE (UNIX and Windows) or DISP=OLD (z/OS) in order to start the SAS IT Resource Management application before using %CPDDUTL with many of the %CPDDUTL control statements. For any tasks (except GENERATE SOURCE) that you perform by using %CPDDUTL, the PDB must exist before the control statements execute.

Additionally, you can use the %CPCAT macro with %CPDDUTL to copy control statements to an entry in a SAS catalog (see "%CPCAT" on page 75). The advantage of

using in-stream text is that it keeps everything together that is related. The advantage of using a separate file is that you can edit the control statements without accidentally editing the rest of the program.

Note: To update the data dictionary in batch mode, you must use the %CPDDUTL macro. Updating the data dictionary by using any method other than the SAS IT Resource Management GUI or the %CPDDUTL macro can result in a corrupted dictionary that cannot be used. △

%CPDDUTL Example

The following examples run the data dictionary utility by using the control statements from a SAS catalog entry.

UNIX Example

```
%cpstart(mode=batch,
         root=pgmlib-location,
         pdb=/your/PDB/,
         access=write);
libname stmts 'your-SAS-DATA-LIB' access=readonly;
%cpddutl(entrynam=stmts.stmtcat.tab1.source);
```

z/OS Example

```
%cpstart(mode=batch,
         root=pgmlib-location,
         pdb=your.PDB,
         disp=old,
         mxglib=mxg.mxg.formats,
         mxgsrc=('mxg.usrid.sourclib' 'mxg.mxg.sourclib')
libname stmts 'your-SAS-data-Lib' disp=shr;
%cpddutl(entrynam=stmts.stmtcat.tab1.source);
```

%CPDDUTL Macro Control Statements

To run the %CPDDUTL macro, write control statements and store them in a SAS catalog source entry, in an external file, or in a member of a PDS. The control statements tell the macro what actions to perform. The control statements enable you to

- create new tables or add supplied tables
- update, delete, or print existing tables
- create, update, delete, or print variables in a table
- generate control statements that will perform some of the previously mentioned tasks, for staged data from a new data source
- load table and variable definitions into the master data dictionary
- perform other miscellaneous tasks.

Additional information about %CPDDUTL is also available in the following documents:

- "%CPDDUTL Syntax Guidelines" on page 568
- "%CPDDUTL Macro Concepts" on page 567

- "%CPDDUTL" on page 569.

Many of the control statements depend on the name of the active table. (For more information about the active table, see the SET TABLE control statement.)

Note: For many of the control statements, the functionality that is provided in batch mode by the %CPDDUTL control statement is also provided interactively in the SAS IT Resource Management GUI. △

%CPDDUTL Statements for Table Definitions

The following control statements enable you to manage tables in the PDB's data dictionary by performing tasks such as "copying" a table definition to the active PDB from the supplied table definitions in the master data dictionary, creating your own table definitions, and printing and maintaining table definitions. Many of the statements for table definitions set or activate a table, which is referred to as the active table.

- "ADD TABLE" on page 573
- "COMPARE TABLES" on page 575
- "CREATE TABLE" on page 587
- "DELETE TABLE" on page 601
- "INSTALL TABLE" on page 611
- "LIST TABLES" on page 614
- "MAINTAIN TABLE" on page 615
- "PRINT TABLE" on page 618
- "PURGE TABLE" on page 619
- "SET TABLE" on page 628
- "STATUS TABLE" on page 629
- "UPDATE TABLE" on page 639
- "XREF TABLE" on page 653

For information about renaming a table, see "%CPRENAME" on page 170.

%CPDDUTL Statements for Regular Variable Definitions

The following control statements enable you to create, delete, list, and update regular variable definitions (but not formula variable definitions or derived variable definitions) for a table. All regular variable control statements apply to the table that is currently active.

- "CREATE VARIABLE" on page 591
- "DELETE VARIABLE" on page 602
- "LIST VARIABLES" on page 615
- "UPDATE VARIABLE" on page 645

For information about renaming regular variables, see "%CPRENAME" on page 170.

%CPDDUTL Statements for Formula Variable Definitions

The following control statements enable you to create, delete, list, and update formula variable definitions in a table. All the statements for formula variable definitions apply to the active table.

- “CREATE FORMULA” on page 584
- “DELETE FORMULA” on page 600
- “LIST FORMULAS” on page 613
- “UPDATE FORMULA” on page 637

For information about renaming formula variables, see “%CPRENAME” on page 170.

%CPDDUTL Statements for Derived Variable Definitions

The following control statements enable you to create, delete, list, and update derived variable definitions in a table. All the statements for derived variable definitions apply to the table that is currently active.

- “CREATE DERIVED” on page 577
- “DELETE DERIVED” on page 600
- “LIST DERIVEDS” on page 613
- “UPDATE DERIVED” on page 631

For information about renaming derived variables, see “%CPRENAME” on page 170.

Additional %CPDDUTL Control Statements

The following control statements enable you to perform miscellaneous tasks such as building views, including additional control statements, and setting default values to be used when creating new table and variable definitions. These statements also enable you to verify the contents of a data dictionary or the syntax of control statements.

- “BUILD VIEWS” on page 574
- “END” on page 603
- “GENERATE SOURCE” on page 604
- “INCLUDE SOURCE” on page 610
- “PRINT SOURCE” on page 617
- “QUIT” on page 620
- “SAVE SOURCE” on page 620
- “SET DEFAULTS” on page 621
- “STOP” on page 630
- “SYNCHRONIZE DICTIONARY” on page 631
- “VERIFY DICTIONARY” on page 652
- “VERIFY SYNTAX” on page 653

ADD TABLE

Adds a supplied table definition to the active PDB’s data dictionary

ADD TABLE Overview

The ADD TABLE control statement “copies” a table definition from the master data dictionary to the data dictionary in the active PDB. To display a list of supplied table definitions for UNIX and Windows environments, select the **`Explore Tables/ Variables`** task on the Administration tab in the GUI for UNIX and Windows. To

display a list of supplied table definitions for z/OS, select **Explore** and then select **All Supplied Tables**.

The ADD TABLE control statement performs the following:

- migrates all of the table definition information for the table, including all of the definition information for its variables, to the active PDB's data dictionary
- copies the list of BY variables (for the detail level) into the lists of class variables for the day, week, month, and year levels
- adds values for any parameters that you set by using the SET DEFAULTS control statement (see "SET DEFAULTS" on page 621)
- applies the table's override control statements, if any
- completes the table by creating, at each level of the PDB, appropriate SAS data sets for storing the table's data at that level.

ADD TABLE Syntax

ADD TABLE
NAME=*table-name*;

Details

NAME=*table-name*
names the table definition to be migrated from the master data dictionary to the active PDB's data dictionary, and names the table to be generated in the PDB. This *table-name* must exist in the master data dictionary, and it must not already exist in the dictionary of the active PDB.

ADD TABLE Notes

When this statement successfully executes, the active table name is set to the table named in this statement. To specify a different table as the active table, see "SET TABLE" on page 628.

The ADD TABLE control statement differs from the CREATE TABLE control statement, which creates an entirely new table definition. The ADD TABLE control statement "copies" an existing table definition from the master data dictionary and adds default values.

Each table definition is associated with one collector. The list of collectors that are supported for a specific operating system is based on the SAS IT Resource Management software license and can affect which tables you can create or add. For more information, refer to your SAS IT Resource Management installation instructions.

BUILD VIEWS

Builds views for one table or all tables

BUILD VIEWS Overview

The BUILD VIEWS control statement builds views and underlying data sets that are associated with the table in the active PDB.

On z/OS, you can optionally build an additional set of views for compatibility with MXG.

When you create or update a table definition, variable definition, or formula variable definition through the SAS IT Resource Management GUI, you can specify whether or not you want the views to be built (created) at that time. If you choose not to create the views, then you may want to create the views later by using this statement.

Views are automatically rebuilt as needed when the %CPDDUTL macro completes its execution, even if you do not specify the BUILD VIEWS control statement. The exception to this rule is that views are not rebuilt when you end %CPDDUTL with a STOP or QUIT control statement instead of an END control statement. In those cases, you need to use the BUILD VIEWS control statement to rebuild views. Additionally, if views are accidentally removed, then you can use the BUILD VIEWS control statement to re-create them.

BUILD VIEWS Syntax

BUILD VIEWS
    NAME=*table-name* | _ALL_;

Details

NAME=*table-name* | _ALL_
is the name of the table for which you want to build views. The table must exist in the active PDB's data dictionary by the time that this statement is encountered. If you specify _ALL_, then views are built for all tables in the PDB.

BUILD VIEWS Notes

The process of building views for tables with many variables can be lengthy.

The BUILD VIEWS control statement has no effect on the table that is currently set as the active table, and the active table is not set to the table that is named in this statement.

COMPARE TABLES

Compares the data dictionary information for two tables

COMPARE TABLES Overview

The COMPARE TABLES control statement compares the data dictionary information for two tables and lists the differences in the SAS log. You can use this statement to compare two different tables in the active PDB, or you can use it to compare a table in the active PDB with a table in the master data dictionary.

For example, when you install a new version of SAS IT Resource Management, the master tables are often updated. You can use the COMPARE TABLES control statement to compare and identify differences between the master table definition and the table definition in the active PDB. You can then use the MAINTAIN TABLE control statement to update the table definition in the active PDB (see "MAINTAIN TABLE" on page 615).

Multiple parameters must be separated by at least one space.

COMPARE TABLES Syntax

COMPARE TABLES
NAME=*table-name*
<,**AGES | NOAGES**>
<,**DATES | NODATES**>
<,**FROM=**PDB | SUPPLIED>
<,**FROM2=**PDB | SUPPLIED>
<,**NAME2=***table-name2*>
<,**TRUNCATE | NOTRUNCATE**>
<,**VARS | NOVARS**>
<,**VER | NOVER**>
<,**VOLATILE | NOVOLATILE**>;

Details

NAME= *table-name*
names the first table in the comparison. This parameter is required. The table name must be specified and must exist in the location specified by the FROM= parameter. (See also the "Notes" section for this control statement.)

AGES | NOAGES
specifies whether or not to compare the age limit information for the specified tables. AGES compares the information and NOAGES ignores the age limit information. *The default is NOAGES*.

DATES | NODATES
specifies whether or not to compare the creation date, last update, and last processing date information in the dictionary. DATES compares this information, and NODATES does not compare this information. *The default is NODATES*.

FROM=PDB | SUPPLIED
specifies where to look for the dictionary information for the specified *table-name*. PDB indicates that the information is located in the active PDB, and SUPPLIED indicates that the information is located in the master data dictionary. *The default is PDB*. (See also the "Notes" section for this control statement.)

FROM2=PDB | SUPPLIED
specifies where to look for the dictionary information for the specified *table-name2*. PDB indicates that the information is located in the active PDB, and SUPPLIED indicates that the information is located in the master data dictionary. *The default is SUPPLIED*. (See also the "Notes" section for this control statement.)

NAME2=*table-name2*
names the second table for this comparison. *The default is the name that is specified for the NAME= parameter.* (See also the "Notes" section for this control statement.)

TRUNCATE | NOTRUNCATE
specifies whether the comparison data that is written to the SAS log should be truncated. When TRUNCATE is specified, the LINESIZE= SAS option should be set to greater than 82 for best results. If NOTRUNCATE is specified, then more of the attributes that are being compared are shown on the SAS log. The SAS LINESIZE= option can be set to a large value to avoid wrapping of lines when they are displayed in the SAS log. *The default is TRUNCATE*.

VARS | NOVARS

specifies whether the variable definitions in the tables should be compared. VARS compares the variable definitions in the specified tables, and NOVARS does not. *The default is VARS*.

VER | NOVER
specifies whether the version of the variables are to be compared. VER means to compare the variables and, NOVER indicates that the version should not be compared. *The default is NOVER*.

VOLATILE | NOVOLATILE
specifies whether or not volatile dictionary information is to be compared. This type of dictionary information includes things such as the current total number of observations. VOLATILE indicates to compare this information, and NOVOLATILE indicates that this information is not to be compared. *The default is NOVOLATILE*.

COMPARE TABLES Notes

This statement does not change either of the tables that are being compared, and the table that is set as the active table is not changed. To specify a different table as the active table, see "SET TABLE" on page 628.

The NAME= and NAME2= parameters can have the same name only if the values of the FROM= and FROM2= parameters are different.

Tables in two different PDBs cannot be compared.

COMPARE TABLES Examples

Example 1

This example compares a table in the active PDB with a supplied table in the master data dictionary. Both tables have the same name. As specified, the versions of the variables in the table and the summary-level age limit information in the dictionary are compared. Because a value for the NAME2= parameter is not specified, the NAME2= parameter defaults to the value of the NAME= parameter and compares two tables with the same name.

```
compare table
             name=TABXYZ ver ages ;
```

Example 2

This example compares the dictionary information for two different tables in the active PDB.

```
compare table
             name=TABXYZ from=PDB
             name2=TAB123 from2=PDB
             novars ;
```

CREATE DERIVED

Creates a derived variable definition for a table

CREATE DERIVED Overview

The CREATE DERIVED control statement creates a derived variable definition in the active table in the active PDB. Derived variables are variables that do not exist in your raw data but that you compute based on your raw data and the values specified by the parameters. Once computed, the values of the derived variables are stored with your raw data in the PDB. Because these variable values are stored in the PDB, these variables can also be used as BY, CLASS, ID, and INDEX variables.

You can also use this statement to create a new derived variable that is similar to an existing derived variable in the same table or another table in the active PDB.

Multiple parameters must be separated by at least one space.

CREATE DERIVED Syntax

CREATE DERIVED
INTERPRET=*interpretation*
NAME=*derived-var-name*
TYPE=NUMERIC | CHARACTER
SOURCE={
source statements
}
<**DAY=**(*statistic-keywords-list*)>
<**WEEK=**(*statistic-keywords-list*)>
<**MONTH=**(*statistic-keywords-list*)>
<**YEAR=**(*statistic-keywords-list*)>
<**DESCRIPTION=**'*string*'>
<**FORMAT=***SAS-format*>
<**ISTATS=**(*statistic-keywords-list*)>
<**KEPT=**YES | NO>
<**LABEL=**'*string*>
<**LENGTH=***number*>
<**SUBJECT=**'*subject-keyword-list*'>
<**VALIDITY=**>
<**WEIGHTVAR=***variable-name*>;

Details

INTERPRET=*interpretation*
indicates the type of information that the variable or derived variable represents. This parameter is required. You can specify one of the following values:

Table 4.1 Values for INTERPRET=

AVERAGE	C2RATE	COUNT
COUNTER	DATE	DATETIME
DECADDRESS	D2RATE	ENUM
FLOAT	FORMULA	GAUGE
HEXFLAGS	INT	IPADDRESS
LABEL	MAXIMUM	MINIMUM

NAME	NETADDRESS	PCTGAUGE
PCTGAUGE100	PERCENT	PERCENT100
RATE	STRING	SUM
TIME	TIMETICKS	UNIXTIME
YNFLAG		

The value can be uppercase or lowercase. For a full description of each value and its defaults, refer to the variable definitions in "Variable Interpretation Types" on page 697. A summary of this information is also available in "Variable Interpretation Type Summary Table" on page 710.

NAME=*derived-var-name*
is the name of the derived variable definition and derived variable to be created. The name must not already exist in the table but it can exist in other tables.

The name of a user-defined variable

- must start with a letter.
- has a maximum length of seven characters (letters and numbers only; do not use underscores).

  The name is not case sensitive; thus, NAME=var1 and NAME=VAR1 are equivalent.

  Note: Underscores are not allowed in names of user-defined variables; however, they might be added automatically, as padding (so that the statistic suffix is in the eighth character) in the variable names that are generated for you when you request predefined statistics at the day, week, month, and year levels on the user-defined variables. △
- must be unique within the table.

  If you might want to use the same variable in another table later, you might also want the name to be unique within all tables in the PDB.

  Similarly, if you might want to install the table in the master data dictionary later, you might also want the name not to match the name of any supplied variable.
- must not be COUNT or TYPE.

TYPE=NUMERIC | CHARACTER
is the type of the variable. The type can be either NUMERIC or CHARACTER. You cannot change the type of the variable once you have created it, unless you delete the variable and create it again.

SOURCE={*source statements*}
specifies the code for the formula variable or derived variable, such as SOURCE={x1=sum(a,b);}. Source statements can be any SAS DATA step statements that result in the creation of a value for the variable that is specified in the NAME= parameter. Remember to end your SAS statements with a semicolon.

Note: Do not omit the braces. If you omit the braces, then you will get various errors. Also, do not make a change in the source code that will cause the variable type to change (such as from numeric to character). △

If you want to create many variables that use the same calculations, then you might want to create a macro that has the common source code for the calculations, add the location of the macro to the SASAUTOS search path, and call the macro in the source statements for each variable.

DAY=(*statistic-keywords-list*)
WEEK=(*statistic-keywords-list*)
MONTH=(*statistic-keywords-list*)
YEAR=(*statistic-keywords-list*)
is the list of statistic calculations that are requested for this regular variable or derived variable at the specified summary levels for the active table. Use blanks to separate the keywords in the list.

You can use the UPDATE TABLE control statement to specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request (in the CREATE VARIABLE control statement) at each level of the table are calculated for each unique set of values of the class variables.

When creating a new variable definition or derived variable definition, specify all statistics that you want to be calculated at each summary level by using the CREATE VARIABLE or CREATE DERIVED control statement, respectively. If you do not specify statistics for a level, then no statistics are kept at that level. If statistics are specified and if the following are true, then the statistics for that variable are calculated:

- The data type for the variable is numeric.
- The table and variable status are set to KEPT=YES.
- The level where the statistics are to be calculated has an age limit greater than zero.
- The variable is not a CLASS or ID variable at the level where the statistics are to be calculated.

When updating a variable definition or derived variable definition, you do not need to respecify statistics that are already being calculated. For example, if you are already keeping the statistics COUNT and SUM and you also want to keep AVERAGE, then you only need to specify AVERAGE in the UPDATE VARIABLE or the UPDATE DERIVED control statement.

The statistic list can contain one or more of these values at each level:

AVERAGE | NOAVERAGE
for the arithmetic mean

COUNT | NOCOUNT
for the accumulated count of the number of observations that have nonmissing values for the variable, starting at 0

CV | NOCV
for the coefficient of variation

MAXIMUM | NOMAXIMUM
for the maximum value

MINIMUM | NOMINIMUM
for the minimum value

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

DESCRIPTION='*string*'
is a 200-character string that describes the variable. The string must be enclosed in single quotation marks. If the string spans multiple lines, then break it in the middle of a word and continue it in the first column of the next line.

FORMAT=*SAS-format*
is a SAS format to be used with this variable. If the variable's value will contain characters, then assign a character format. If the variable's value will contain numbers, then assign a numeric format. These formats are used for all presentation of the data for this variable.

For more information on default formats that are related to the interpretation type, see the INTERPRET= parameter on the CREATE VARIABLE or CREATE DERIVED control statement. Also see "Variable Interpretation Types" on page 697. For more information on SAS formats, refer to *SAS Language Reference* for your current release of SAS.

ISTATS=(*statistic-keywords-list*)
lists the default statistics for a variable or derived variable. The statistics are saved with the variable definition, but they are not calculated within the active table. These statistics are used when you run INSTALL TABLE or ADD TABLE control statements, because the master data dictionary does not keep reduction level statistics information.

When called by the UPDATE VARIABLE control statement or the UPDATE DERIVED control statement, ISTATS= applies changes to the default statistics. If an initial statistic is not specified using the UPDATE VARIABLE control statement, the existing setting for that statistic is not changed.

When called by the SET DEFAULTS control statement, ISTATS= sets the default statistics for a variable if you do not set the default statistics when you create the variable.

You can specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request at each level are calculated for each class variable group in each summary level.

The statistics that are calculated for the active table are those that are specified by the DAY=, WEEK=, MONTH=, and YEAR= parameters. Therefore, when you install a table in the master data dictionary, it stores the ISTATS with the table definition. When you later add that table to another PDB, the ADD TABLE statement populates the DAY=, WEEK=, MONTH=, and YEAR= statistics based on the ISTATS= list of statistics. You can then customize the added table by using the SAS IT Resource Management GUI or an UPDATE VARIABLE/UPDATE DERIVED statement with the DAY=, WEEK=, MONTH=, and YEAR= parameters.

AVERAGE | NOAVERAGE
for the arithmetic mean for all observations

COUNT | NOCOUNT

for an accumulated count of the number of observations that have nonmissing values, starting at zero

CV | NOCV
for the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAXIMUM | NOMAXIMUM
for the maximum value for all observations

MINIMUM | NOMINIMUM
for the minimum value for all observations

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values for all observations

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: ISTATS= can be specified on the SET DEFAULTS, CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, and UPDATE DERIVED statements. If you install or add a table for which ISTATS have not been specified, then the table is given the default statistic values for ISTATS=, which have been specified in SET DEFAULTS for the current set of %CPDDUTL statements. If default values have not been set via SET DEFAULTS, then the variable is treated as if all the statistics in the ISTATS= list were set to NO. If the table is installed and then added, then the added table does not keep any statistics at the day, week, month, or year level unless you set them by using an UPDATE statement. △

If you use a SET DEFAULTS statement for a single variable, then a useful default setting is one that is based on the variable's interpretation. (For more about the relation between interpretation and initial statistics, see "GENERATE SOURCE" on page 604.)

If the SET DEFAULTS statement will apply to a number of variables, then a useful default setting is ISTATS=(AVERAGE).

KEPT=YES | NO
indicates to the process step or reduce step whether data for this variable is to be kept in the PDB. If you specify NO, then the variable's definition is retained in the dictionary, but existing data is deleted and no new data for this variable is processed or reduced into the PDB. *The default value is NO*.

Note: If you have no current use for the variable, but do anticipate future use, you might want to set the variable's Kept status to *No*. When the variable's Kept status is set to *No*, the variable's definition is retained but ignored, and the variable's values, if any, are discarded. (There will be missing values in all

existing observations for the variables that had their Kept status changed from *Yes* to *No*.) △

LABEL='*string*'
is a 40-character string that describes the data in the variable at the detail level. The string must be enclosed in single quotation marks. In the day, week, month, and year levels, the labels are formed automatically by concatenating the statistic name, a period, and the string.

LENGTH=*number*
is the maximum length of the variable as stored in the PDB. *The default value is 0*. Character variables can be as long as 200 characters. The minimum length of a numeric variable is three bytes and its maximum length is eight bytes.

Note: On z/OS, a numeric variable can have a length of two, but that length is not supported in any other operating environments. Therefore, if you specify a length of two, then you cannot report on the data from the SAS IT Resource Management client. △

SUBJECT='*subject-keyword-list*'
specifies a list of keywords that can be used to categorize this variable. The maximum length of the list is 200 characters. Use SUBJECT=' ' in order to leave the list blank. You must use blanks to separate keywords in the list, and the list must be enclosed in single quotation marks.

When updating the SUBJECT= parameter list, you must specify all keywords that you want to use at the time that you update the list, because this parameter does not use pre-existing values.

VALIDITY=
is not currently used.

WEIGHTVAR=*variable-name*
names the alternate variable to be used for statistical weighting. The variable must already exist in the table.

If you do not specify this parameter, then default variable weighting and normalization are used. *By default, analysis variables in INTERVAL tables are weighted and normalized by the DURATION variable during report-time summarization, such as the summarization when Summary Time Period is not AS IS. By default, analysis variables in EVENT tables are not weighted or normalized.*

The following variable interpretation types are normalized (that is, converted to rates for comparison purposes) if the table is an INTERVAL table:

- COUNT
- TIME
- TIMETICKS.

The following variable interpretation types are not normalized (converted) if the table is an INTERVAL table:

- GAUGE
- INT
- MAXIMUM
- MINIMUM.

All other numeric variable interpretation types are weighted if the table is an INTERVAL table.

By default, no weighting or normalization occurs for variables in EVENT tables unless the WEIGHTVAR= parameter is used to specify a weight variable. If a weight variable is specified for a variable in an EVENT table, then the same normalization or weighting rules that are described above are used.

See also “Variable Interpretation Types” on page 697.

CREATE DERIVED Notes

For more information about the order in which derived and formula variables are calculated, see the section “How to Define Derived and Formula Variables” in the chapter “Administration: Working with Variables” in the *SAS IT Resource Management User's Guide*

Some of the regular and/or derived variables in the table can be used as BY, CLASS, ID, or INDEX variables. These variable roles are assigned by UPDATE TABLE, not by CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, or UPDATE DERIVED.

CREATE FORMULA

Creates a definition for a new formula variable in a table

CREATE FORMULA Overview

The CREATE FORMULA control statement creates the definition for a new formula variable in the active table in the active PDB. Data values for the formula variable are not stored in the PDB, but they are calculated when they are requested, such as when a report is generated. Formula variables cannot be indexed.

Multiple parameters must be separated by at least one space.

CREATE FORMULA Syntax

CREATE FORMULA
LEVELS=’<**DETAIL**><**DAY**><**WEEK**><**MONTH**><**YEAR**>’
NAME=*formula-var-name*
SOURCE={
source statements
}
TYPE=NUMERIC | CHARACTER
<**DESCRIPTION=** *’string’*>
<**FORMAT=***SAS-format*>
<**KEPT=**YES | NO>
<**LABEL=***’string’*>
<**LENGTH=***number*>
<**SUBJECT=***’subject-keyword-list’*>;

Details

LEVELS=’*levels*’
is a list of levels (DETAIL, DAY, WEEK, MONTH, YEAR) at which the formula variable is to be available. If the formula variable relies on other regular, derived, or formula variables, then those variables and formula variables must exist at all the specified levels. The levels must be enclosed in single quotation marks and separated by one or more spaces, as in LEVELS=’DETAIL DAY’.

Note: The formula variable itself, not a statistic that is based on the formula variable, is available at the specified levels. △

NAME=*formula-var-name*
is the name of the formula variable "definition" and formula variable to be created. The name must not already exist in the table but it can exist in other tables. Do not use the name COUNT or the name TYPE.

The name of a user-defined variable

- must start with a letter.
- has a maximum length of seven characters (letters and numbers only; do not use underscores).

  The name is not case sensitive; thus, NAME=var1 and NAME=VAR1 are equivalent.

  Note: Underscores are not allowed in names of user-defined variables; however, they might be added automatically, as padding (so that the statistic suffix is in the eighth character) in the variable names that are generated for you when you request predefined statistics at the day, week, month, and year levels on the user-defined variables. △

- must be unique within the table.

  If you might want to use the same variable in another table later, you might also want the name to be unique within all tables in the PDB.

  Similarly, if you might want to install the table in the master data dictionary later, you might also want the name not to match the name of any supplied variable.
- must not be COUNT or TYPE.

SOURCE={*source statements*}
specifies the code for the formula variable or derived variable, such as SOURCE={x1=sum(a,b);}. Source statements can be any SAS DATA step statements that result in the creation of a value for the variable that is specified in the NAME= parameter. Remember to end your SAS statements with a semicolon.

Note: Do not omit the braces. If you omit the braces, then you will get various errors. Also, do not make a change in the source code that will cause the variable type to change (such as from numeric to character). △

If you want to create many variables that use the same calculations, then you might want to create a macro that has the common source code for the calculations, add the location of the macro to the SASAUTOS search path, and call the macro in the source statements for each variable.

TYPE=NUMERIC | CHARACTER
is the type of the formula variable. The type can be either NUMERIC or CHARACTER. You cannot change the type of the formula variable after you create it, unless you delete the formula variable and create it again.

DESCRIPTION='*string*'
is a 200-character string that provides information about the formula variable. The string must be enclosed in single quotation marks. If the string spans multiple lines, break it in the middle of a word and continue it in the first column of the next line.

FORMAT=*SAS-format*
is a SAS format to be used with this variable. If the variable's value will contain characters, then assign a character format. If the variable's value will contain numbers, then assign a numeric format. These formats are used for all presentation of the data for this variable.

For more information on default formats that are related to the interpretation type, see the INTERPRET= parameter on the CREATE VARIABLE or CREATE DERIVED control statement. Also see "Variable Interpretation Types" on page 697. For more information on SAS formats, refer to *SAS Language Reference* for your current release of SAS.

KEPT=YES | NO
indicates whether this formula variable is used or ignored. If you specify NO, then the variable's definition is retained in the dictionary, but values are not calculated. If you specify YES (the default), the variable's values are calculated in the view.

LABEL='*string*'
is a 40-character string that describes the formula variable. The string must be enclosed in single quotation marks. This label is used in the view that contains this formula variable.

LENGTH=*number*
is the maximum length of the variable as stored in the PDB. *The default value is 0*. Character variables can be as long as 200 characters. The minimum length of a numeric variable is three bytes and its maximum length is eight bytes.

Note: On z/OS, a numeric variable can have a length of two, but that length is not supported in any other operating environments. Therefore, if you specify a length of two, then you cannot report on the data from the SAS IT Resource Management client. △

SUBJECT='*subject-keyword-list*'
specifies a list of keywords that can be used to categorize this variable. The maximum length of the list is 200 characters. Use SUBJECT=' ' in order to leave the list blank. You must use blanks to separate keywords in the list, and the list must be enclosed in single quotation marks.

When updating the SUBJECT= parameter list, you must specify all keywords that you want to use at the time that you update the list, because this parameter does not use pre-existing values.

CREATE FORMULA Notes

The active table must be set before you can run this statement. To specify the active table, see "SET TABLE" on page 628.

Formula variables cannot have default statistics at the day, week, month, or year level, because formula variables are created as needed, and no data is kept at the levels. If you want statistics at the summary levels, then you must create additional formula variables (at each level for which you want statistics) that are themselves statistics.

When you create the new formula variable, you can use the _LEVEL_ variable to determine the active level of the PDB, as shown in this example:

```
select(_level_);
   when ('DETAIL')  formula=3.14;
   when ('DAY')     formula=2.2;
   when ('WEEK')    formula=1.77;
   when ('MONTH')   formula=1;
   when ('YEAR')    formula=0;
   end;
```

LEVEL can also be used in an IF statement. The following example tests _LEVEL_ for a formula variable called *xxx*:

```
if _level_='DETAIL' then
     xxx=yyy/zzz;
```

```
else
     xxx=yyy____s/zzz____s;
Note: the number of underscores is four for each variable.
```

You can use level-specific variables, such as CPUTM__S, in a formula variable. However, because the same source code is used in all the views for the table, any variables that are referenced in the source code could cause you to receive the message "NOTE: CPUTM__S is uninitialized." This is a problem only if the variable is used to calculate the formula at that level of the table. (If the formula variable relies on other regular, derived, or formula variables, then these variables and formula variables must exist at all the specified levels.)

In addition to being calculated as needed, formula variables are created in alphabetical order in the view after all other regular and derived variables have been defined. Therefore, the definition of a formula variable can refer to another formula variable in the same table, as long as the variable that you refer to is listed alphabetically before the variable in which it is used. For example, a variable that is named A could be used within the calculations of the value for a variable that is named B, because A comes before B alphabetically. Or you can define one formula variable and, in its SOURCE= parameter, define intermediate variables in any order; that is, they do not have to be in alphabetical order. For example, you might use the following:

```
SOURCE={c=var1 + var2;
        b= var3;
        a=c/b;
       }
```

For more information about the order in which derived and formula variables are calculated, see the section "How to Define Derived and Formula Variables." in the chapter "Administration: Working with Variables" in the *SAS IT Resource Management User's Guide*.

CREATE TABLE

Creates a new table definition in the active PDB's data dictionary

CREATE TABLE Overview

The CREATE TABLE control statement creates a new user-defined table definition in the active PDB's data dictionary, and in doing so it also automatically creates the required variable definitions for DATETIME, LSTPDATE, and DURATION. (DURATION is created for TYPE=INTERVAL tables only.) This control statement implements the table by initializing at each level of the PDB one or more empty SAS data sets for storing the table's data.

Following the CREATE TABLE control statement, you can add additional variables and formula variables with the CREATE VARIABLE, CREATE FORMULA, and CREATE DERIVED control statements. You should use the UPDATE TABLE control statement to create the table's BY, CLASS, ID, and INDEX variable lists. And you can use the UPDATE TABLE control statement stand to make any adjustments to the values of the COLLECTOR=, TOOLNAME=, and EXTNAME= parameters.

Refer to the control statement "SET DEFAULTS" on page 621 for more information on the default values that apply if you do not specify all the parameters.

Note: Availability tables are created by the %CPAVAIL macro, not the CREATE TABLE control statement. For more information, see "%CPAVAIL" on page 64. △

Note: Multiple parameters must be separated by at least one space. △

CREATE TABLE Syntax

CREATE TABLE
NAME=*table-name*
TYPE=INTERVAL | EVENT
<**ARCHIVE=**YES | NO>
<**COLLECTOR=***collector* | *generic*>
<**DESCRIPTION=**'*string*'>
<**DETAIL=**(**AGELIMIT=***number*)>
<**DAY=**(**AGELIMIT=***number*)>
<**WEEK=**(**AGELIMIT=***number*)>
<**MONTH=**(**AGELIMIT=***number*)>
<**YEAR=**(**AGELIMIT**=*number*)>
<**EXTNAME=***name*>
<**IDNUM=***number*>
<**KEPT=**YES | NO>
<**LABEL=**'*string*'>
<**TOOLNAME=***name*>;

Details

NAME=*table-name*
is the name of the table definition and table to be created. The name of a user-defined table

- must start with a letter. The letter *U* is strongly recommended for user-defined tables.
- has a maximum length of seven characters (U plus zero to six letters, numbers, and/or underscores).

  The name is not case sensitive; thus, NAME=utable1 and NAME=UTABLE1 are equivalent.
- must be unique within the PDB.

  If you might want to use the same table in another PDB later, you might also want the name to be unique within all the PDBs at your site.

  Similarly, if you might want to install the table in the master data dictionary later, you might also want the name not to match the name of any supplied table.
- must avoid the name space of any table within the PDB. The name space of a table consists of the name itself, the name with *D* appended, and the name with *P* appended. For example, if a table name is *CPU*, then the table uses the names *CPU*, *CPUD*, and *CPUP*. Thus, you must avoid all three of these names in choosing a name for a new table.

  For the reasons that were mentioned above, you might also want to avoid the name space of tables in other PDBs at your site and the name space of supplied tables.

  For more information about the table name with *D* and *P* appended, see the section "Overview of Tables" in the chapter "Administration: Working with Tables" in the *SAS IT Resource Management User's Guide*.

TYPE= INTERVAL | EVENT

is the type of data in the table. Reduction statistics are calculated differently, based on the table type. For more information about the calculations, refer to the control statement "CREATE VARIABLE" on page 591 . You cannot change the table type once it has been created, unless you delete the table definition and then create it again. However, when you delete the table definition, you will also delete any data in the table.

Valid values for TYPE are:

INTERVAL
indicates that the data in this table is collected at specifically timed intervals, that is, synchronously. The interval can be regular or irregular. The DURATION variable is automatically created for interval tables. The DURATION variable contains the amount of time in seconds, during which the data was collected, typically starting at the time in the DATETIME variable.

EVENT
indicates that the data in this table is collected in response to specific events instead of at regular intervals, that is, asynchronously.

The DURATION variable cannot exist in this type of data. Reduction will fail if the DURATION variable exists in an event table. The time of the specific event is recorded in the DATETIME variable.

For any given table, the data must be either interval data or event data, but not both. If you have both types of data, then use a separate table for each type. For example, you might use TYPE=INTERVAL to record network traffic in intervals of every 15 minutes, and you might use TYPE=EVENT to record events such as the completion of a task.

ARCHIVE=YES | NO
indicates whether or not your new data for this table should be archived when the data is processed into the PDB. If you want to archive your data when it is processed into the PDB, then specify YES. Archiving of data is "turned on" when you specify ARCHIVE=YES for a table. If ARCHIVE=YES, then when data is processed into this table, the data is also archived.

If you do not want your data archived, then specify NO. *This parameter is not required and the default value is NO*.

COLLECTOR=*collector-name* | *generic*
is the name that (along with the tool name) enables the process task to access and run the correct staging code and use the correct table and variable definitions. The value of the collector name specifies the name of the data collector or data source that creates the log file data. (Which data collectors or data sources are supported by a host depends on the SAS IT Resource Management software license. For more information, see the SAS IT Resource Management installation instructions.) This is the value that you specify for the COLLECTR= parameter on the process macro (%CxPROCES) when you process your data into the PDB.

Note: %CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES. △

For more about the value to use for *collector-name*, see the Notes section for the UPDATE TABLE control statement.

If the name of the collector contains special characters, then enclose the name in single quotation marks. Here is an example, if you are using HP-OVPA:

```
collector='hp-ovpa'
```

DESCRIPTION='*string*'
is a 200-character string that you can use to describe the data in the table. The string must be enclosed in single quotation marks. If the string spans multiple

lines, break it in the middle of a word and continue in the first column of the next line.

DETAIL=(AGELIMIT=*number*)
DAY=(AGELIMIT=*number*)
WEEK=(AGELIMIT=*number*)
MONTH=(AGELIMIT=*number*)
YEAR=(AGELIMIT=*number*)

defines the age range that the data can have at each PDB level. Existing data in each level of the PDB is "aged out" (removed) when the range of DATETIME reaches the value that is specified in the AGELIMIT= parameter. The default value and representation of the number value at each level are as follows:

- DETAIL - *number* represents days; the default is 10 days.
- DAY - *number* represents days; the default is 45 days.
- WEEK - *number* represents weeks; the default is 15 weeks.
- MONTH - *number* represents months; the default is 18 months.
- YEAR - *number* represents years; the default is 5 years.

The *number* must be an integer.

Note: If AGELIMIT=0 at the detail level, then your data is stored in detail level until the next reduce or process task runs. Therefore, if the reduce task runs immediately after the process task, then the data is processed, reduced into the summary levels, and removed from the detail level. However, if the process task runs again before the data has been reduced, then the existing data in the detail level is overwritten and, thus, not incorporated in the statistics at the summary levels. △

Using AGELIMIT=0 at the detail level is useful if you are collecting and processing large amounts of data on a daily basis; however, to avoid losing data at summary levels, always reduce the data immediately after you process it. Using AGELIMIT=0 at any other level means that no data is kept at that level.

Note: At the week, month, or year level, if AGELIMIT=*number*, then data is kept for the most recent partial week, month, or year and for the previous *number–1* full weeks, months, or years. In other words, AGELIMIT=*number* means up to but not including *number* full weeks, months, or years of data. △

For information about setting age limits for your data, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide* .

EXTNAME=*name*

is the name of the external data source from which data is processed into the detail level of the PDB. The data source could be a SAS data set or an external file, depending on the data collector or data source from which you obtain the data.

For more about the value to use for external-name, see the Notes section for the UPDATE TABLE control statement.

The name can be a maximum of eight characters long on z/OS or 200 characters long on other operating environments. The name must begin with a letter, can contain letters and numbers, and can contain blanks or special characters. The name is not case sensitive.

If you do not specify this parameter, then a blank value is stored in the data dictionary. The variable name field is used to determine what variable is to be processed. That is, when the next process task runs, the *blank* external name defaults to the value that is specified in the NAME= parameter.

IDNUM=*number*

specifies a numeric identifier that has different uses for each collector. Refer to the information about a specific collector in the *SAS IT Resource Management Server Setup Guide*.

This parameter is used only for UNIX and Windows.

KEPT=YES | NO
indicates to the process task whether this table is to be used or ignored. If you specify YES, then the table definition and existing data at any level of the PDB are retained. New data for this table is processed into the detail level of the PDB, and any existing data in the detail level is processed into the summary levels of the PDB. If you specify NO, then the table definition and existing data at any level of the PDB are retained in the data dictionary, but new data for this table is not processed into the detail level of the PDB and any existing data in the detail level is not processed to the summary levels of the PDB. *The default value is NO.*

The KEPT= parameter is ignored when you specify a list of tables in any of the %C*x*PROCESS macros, where *x* is M, P, S, or W.

LABEL='*string*'
is a 40-character string describing the data in the table. The string must be enclosed in single quotation marks. This label is used in the views created for this table.

TOOLNAME=*name*
is the name that (along with the collector-name) enables the process task to access and run the correct staging code and to use the correct table and variable definitions.

For more about the value to use for tool-name, see the Notes section for the UPDATE TABLE control statement.

This name cannot exceed 32 characters.

CREATE TABLE Notes

When this statement successfully completes its execution, the active table is set to the table named in this statement. To specify a different table as the active table, see "SET TABLE" on page 628.

If you want to copy an existing table definition from the master data dictionary, then use the ADD TABLE statement.

To create a table like another table (similar to the functionality of the LIKE= parameter for variables), use the GENERATE SOURCE statement.

The list of collectors that are supported for a specific operating environment is based on the SAS IT Resource Management software license and might affect which tables you can create or add. For more information, refer to your SAS IT Resource Management installation instructions.

For more information on creating a table from MIB (Management Information Base) definitions, see the chapter "Setup: The MIB to Dictionary Compiler" in the *SAS IT Resource Management User's Guide*

For information on setting age limits for your data, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*.

CREATE VARIABLE

Creates a new regular variable definition in a table

CREATE VARIABLE Overview

The CREATE VARIABLE control statement creates the definition for a new regular variable in the active table in the active PDB.

You can also use this statement to create a new regular variable that is similar to an existing regular variable in the same table or another table in the active PDB.

Multiple parameters must be separated by at least one space.

CREATE VARIABLE Syntax

CREATE VARIABLE
INTERPRET=*interpretation*
NAME=*variable-name*
TYPE=NUMERIC | CHARACTER
<**DAY=**(*statistic-keywords-list*)>
<**WEEK=**(*statistic-keywords-list*)>
<**MONTH=**(*statistic-keywords-list*)>
<**YEAR=**(*statistic-keywords-list*)>
<**DESCRIPTION=**'*string*'>
<**EXTNAME=***name*>
<**FORMAT=***SAS-format*>
<**IDNUM=***number*>
<**INFORMAT=**>
<**ISTATS=**(*statistic-keywords-list*)>
<**KEPT=**YES | NO>
<**LABEL=**'*string*'>
<**LENGTH=***number*>
<**LIKE=**'*like-name*'>
<**OID=***oid-specification*>
<**SUBJECT=**'*subject-keyword-list*'>
<**VALIDITY=**>
<**WEIGHTVAR=***variable-name*>;

Details

INTERPRET=*interpretation*
indicates the type of information that the variable or derived variable represents. This parameter is required. You can specify one of the following values:

Table 4.2 Values for INTERPRET=

AVERAGE	C2RATE	COUNT
COUNTER	DATE	DATETIME
DECADDRESS	D2RATE	ENUM
FLOAT	FORMULA	GAUGE
HEXFLAGS	INT	IPADDRESS
LABEL	MAXIMUM	MINIMUM
NAME	NETADDRESS	PCTGAUGE
PCTGAUGE100	PERCENT	PERCENT100

RATE	STRING	SUM
TIME	TIMETICKS	UNIXTIME
YNFLAG		

The value can be uppercase or lowercase. For a full description of each value and its defaults, refer to the variable definitions in "Variable Interpretation Types" on page 697. A summary of this information is also available in "Variable Interpretation Type Summary Table" on page 710.

NAME=*variable-name*
is the name of the regular variable definition and regular variable to be created. The name must not already exist in the table, but it can exist in other tables.

The name of a user-defined variable

- must start with a letter.
- has a maximum length of seven characters (letters and numbers only; do not use underscores).

  The name is not case sensitive; thus, NAME=var1 and NAME=VAR1 are equivalent.

  Note: Underscores are not allowed in names of user-defined variables; however, they might be added automatically, as padding (so that the statistic suffix is in the eighth character) in the variable names that are generated for you when you request predefined statistics at the day, week, month, and year levels on the user-defined variables. △

- must be unique within the table.

  If you might want to use the same variable in another table later, you might also want the name to be unique within all tables in the PDB.

  Similarly, if you might want to install the table in the master data dictionary later, you might also want the name not to match the name of any supplied variable.
- must not be COUNT or TYPE.

TYPE=NUMERIC | CHARACTER
is the type of the variable. The type can be either NUMERIC or CHARACTER. You cannot change the type of the variable once you have created it, unless you delete the variable and create it again.

DAY=(*statistic-keywords-list*)
WEEK=(*statistic-keywords-list*)
MONTH=(*statistic-keywords-list*)
YEAR=(*statistic-keywords-list*)
is the list of statistic calculations that are requested for this regular variable or derived variable at the specified summary levels for the active table. Use blanks to separate the keywords in the list.

You can use the UPDATE TABLE control statement to specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request (in the CREATE VARIABLE control statement) at each level of the table are calculated for each unique set of values of the class variables.

When creating a new variable definition or derived variable definition, specify all statistics that you want to be calculated at each summary level by using the CREATE VARIABLE or CREATE DERIVED control statement, respectively. If you

do not specify statistics for a level, then no statistics are kept at that level. If statistics are specified and if the following are true, then the statistics for that variable are calculated:

- The data type for the variable is numeric.
- The table and variable status are set to KEPT=YES.
- The level where the statistics are to be calculated has an age limit greater than zero.
- The variable is not a CLASS or ID variable at the level where the statistics are to be calculated.

When updating a variable definition or derived variable definition, you do not need to respecify statistics that are already being calculated. For example, if you are already keeping the statistics COUNT and SUM and you also want to keep AVERAGE, then you only need to specify AVERAGE in the UPDATE VARIABLE or the UPDATE DERIVED control statement.

The statistic list can contain one or more of these values at each level:

AVERAGE | NOAVERAGE
for the arithmetic mean

COUNT | NOCOUNT
for the accumulated count of the number of observations that have nonmissing values for the variable, starting at 0

CV | NOCV
for the coefficient of variation

MAXIMUM | NOMAXIMUM
for the maximum value

MINIMUM | NOMINIMUM
for the minimum value

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

DESCRIPTION='*string*'
is a 200-character string that describes the variable. The string must be enclosed in single quotation marks. If the string spans multiple lines, then break it in the middle of a word and continue it in the first column of the next line.

EXTNAME=*name*

is the name of the variable as it appears in the external data source from which data is processed into the detail level of the PDB. The data source could be a SAS data set or an external file, depending on the collector that you used to collect the data.

If COLLECTOR=GENERIC for this table, then EXTNAME= must be the name of the variable as it exists in the SAS data set or view. The variable name must be a valid SAS name. It can be a maximum of eight characters long, must begin with a letter, can contain letters and numbers, and cannot contain blanks or special characters. The name is not case sensitive.

If you do not specify this parameter, then a blank value is stored in the data dictionary. The variable name field is used to determine what variable is to be processed. That is, when the next process task runs, the *blank* external name defaults to the value that is specified in the NAME= parameter.

FORMAT=*SAS-format*
is a SAS format to be used with this variable. If the variable's value will contain characters, then assign a character format. If the variable's value will contain numbers, then assign a numeric format. These formats are used for all presentation of the data for this variable.

For more information on default formats that are related to the interpretation type, see the INTERPRET= parameter on the CREATE VARIABLE or CREATE DERIVED control statement. Also see "Variable Interpretation Types" on page 697. For more information on SAS formats, refer to *SAS Language Reference* for your current release of SAS.

IDNUM=*number*
specifies a numeric identifier that has different uses for each collector. Refer to the information about a specific collector in the *SAS IT Resource Management Server Setup Guide*.

This parameter is used only for UNIX and Windows.

INFORMAT=
is not currently used.

ISTATS=(*statistic-keywords-list*)
lists the default statistics for a variable or derived variable. The statistics are saved with the variable definition, but they are not calculated within the active table. These statistics are used when you run INSTALL TABLE or ADD TABLE control statements, because the master data dictionary does not keep reduction level statistics information.

When called by the UPDATE VARIABLE control statement or the UPDATE DERIVED control statement, ISTATS= applies changes to the default statistics. If an initial statistic is not specified using the UPDATE VARIABLE control statement, the existing setting for that statistic is not changed.

When called by the SET DEFAULTS control statement, ISTATS= sets the default statistics for a variable if you do not set the default statistics when you create the variable.

You can specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request at each level are calculated for each class variable group in each summary level.

The statistics that are calculated for the active table are those that are specified by the DAY=, WEEK=, MONTH=, and YEAR= parameters. Therefore, when you install a table in the master data dictionary, it stores the ISTATS with the table definition. When you later add that table to another PDB, the ADD TABLE statement populates the DAY=, WEEK=, MONTH=, and YEAR= statistics based

on the ISTATS= list of statistics. You can then customize the added table by using the SAS IT Resource Management GUI or an UPDATE VARIABLE/UPDATE DERIVED statement with the DAY=, WEEK=, MONTH=, and YEAR= parameters.

AVERAGE | NOAVERAGE
for the arithmetic mean for all observations

COUNT | NOCOUNT
for an accumulated count of the number of observations that have nonmissing values, starting at zero

CV | NOCV
for the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAXIMUM | NOMAXIMUM
for the maximum value for all observations

MINIMUM | NOMINIMUM
for the minimum value for all observations

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values for all observations

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: ISTATS= can be specified on the SET DEFAULTS, CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, and UPDATE DERIVED statements. If you install or add a table for which ISTATS have not been specified, then the table is given the default statistic values for ISTATS=, which have been specified in SET DEFAULTS for the current set of %CPDDUTL statements. If default values have not been set via SET DEFAULTS, then the variable is treated as if all the statistics in the ISTATS= list were set to NO. If the table is installed and then added, then the added table does not keep any statistics at the day, week, month, or year level unless you set them by using an UPDATE statement. △

If you use a SET DEFAULTS statement for a single variable, then a useful default setting is one that is based on the variable's interpretation. (For more about the relation between interpretation and initial statistics, see "GENERATE SOURCE" on page 604.)

If the SET DEFAULTS statement will apply to a number of variables, then a useful default setting is ISTATS=(AVERAGE).

KEPT=YES | NO
indicates to the process step or reduce step whether data for this variable is to be kept in the PDB. If you specify NO, then the variable's definition is retained in the

dictionary, but existing data is deleted and no new data for this variable is processed or reduced into the PDB. *The default value is NO*.

Note: If you have no current use for the variable, but do anticipate future use, you might want to set the variable's Kept status to *No*. When the variable's Kept status is set to *No*, the variable's definition is retained but ignored, and the variable's values, if any, are discarded. (There will be missing values in all existing observations for the variables that had their Kept status changed from *Yes* to *No*.) △

LABEL='*string*'
is a 40-character string that describes the data in the variable at the detail level. The string must be enclosed in single quotation marks. In the day, week, month, and year levels, the labels are formed automatically by concatenating the statistic name, a period, and the string.

LENGTH=*number*
is the maximum length of the variable as stored in the PDB. *The default value is 0*. Character variables can be as long as 200 characters. The minimum length of a numeric variable is three bytes and its maximum length is eight bytes.

Note: On z/OS, a numeric variable can have a length of two, but that length is not supported in any other operating environments. Therefore, if you specify a length of two, then you cannot report on the data from the SAS IT Resource Management client. △

LIKE='*like-name*'
specifies the location of the existing variable from which you want to create the new variable. This parameter has three parts.

Example: 'SUPPLIED.MYTAB.NEWVAR' or 'NEWVAR'.

The first part of the parameter is the location of the data dictionary. The value is SUPPLIED (for the master data dictionary) or PDB (for the active PDB). This first part is optional. If it is specified, then it must be followed by a period (.). *If it is omitted, then PDB is used.*

The second part of the parameter is a valid table name within the location that is specified in the first part. This second part is optional, but it must be specified if the first part is specified. If the second part is specified, then a period (.) must follow the name. *If it is not specified, then the active table from the active PDB is used.*

The third part of the parameter is a variable name that must exist in the table that is specified by the first and second parts of the parameter. *The third part is not optional*.

The new variable is created with exactly the same attributes as the variable named in the *like-var-name*.

When the LIKE= parameter is used, no other parameter is active.

OID=*oid-specification*
is an SNMP OID (object ID) of the form *n.n.n.n.n.n*. The value must be unique for the variable, if you specify a value. Use *OID=.* to specify none.

This parameter is used only on UNIX.

SUBJECT='*subject-keyword-list*'
specifies a list of keywords that can be used to categorize this variable. The maximum length of the list is 200 characters. Use SUBJECT=' ' in order to leave the list blank. You must use blanks to separate keywords in the list, and the list must be enclosed in single quotation marks.

When updating the SUBJECT= parameter list, you must specify all keywords that you want to use at the time that you update the list, because this parameter does not use pre-existing values.

VALIDITY=
is not currently used.

WEIGHTVAR=*variable-name*
names the alternate variable to be used for statistical weighting. The variable must already exist in the table.

If you do not specify this parameter, then default variable weighting and normalization are used. *By default, analysis variables in INTERVAL tables are weighted and normalized by the DURATION variable during report-time summarization, such as the summarization when Summary Time Period is not AS IS. By default, analysis variables in EVENT tables are not weighted or normalized.*

The following variable interpretation types are normalized (that is, converted to rates for comparison purposes) if the table is an INTERVAL table:

- COUNT
- TIME
- TIMETICKS.

The following variable interpretation types are not normalized (converted) if the table is an INTERVAL table:

- GAUGE
- INT
- MAXIMUM
- MINIMUM.

All other numeric variable interpretation types are weighted if the table is an INTERVAL table.

By default, no weighting or normalization occurs for variables in EVENT tables unless the WEIGHTVAR= parameter is used to specify a weight variable. If a weight variable is specified for a variable in an EVENT table, then the same normalization or weighting rules that are described above are used.

See also "Variable Interpretation Types" on page 697.

CREATE VARIABLE Notes

The DATETIME, LSTPDATE, and DURATION variables are automatically created. If you create SHIFT and HOUR variables, then they will be automatically calculated, based on the DATETIME variable for the observations and the shift matrix that you defined by using the process macro (%CSPROCES, %CMPROCES, %CWPROCES, or %CPPROCES).

You can create a variable with attributes similar to those of another variable by specifying the LIKE= and NAME= parameters on the CREATE VARIABLE control statement. *When you specify the LIKE= parameter, you must specify the NAME= parameter, but you cannot specify any other required or optional parameters or you will get error messages.* All other parameters that are currently set for the variable that is specified in the LIKE= parameter are applied to the variable that is specified in the NAME= parameter.

After you create the new variable, you can use the UPDATE VARIABLE control statement to update the new variable in the data dictionary. Specifically, you should change the value of the EXTNAME= parameter for the new variable in order to avoid conflicts with the existing variable that has that external name.

Some of the regular and/or derived variables in the table can be used as BY, CLASS, ID, or INDEX variables. These variable roles are assigned by UPDATE TABLE, not by CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, or UPDATE DERIVED.

CREATE VARIABLE Examples

Example 1

This example creates a variable that is like a variable in a different table in the active PDB and then changes the description and external name for the new variable.

```
set table
  name=TABXYZ ;            /* set active table */
create variable                    /* create new variable */
  name=CPUBUSY
  like='PDB.TABLEABC.PCTCPUBY' ;   /* like this variable */
update variable                    /* update new variable */
  name=CPUBUSY
  description='Total CPU percent busy'       /* override */
  extname=newextname ;        /*changes external name of new variable*/
```

Example 2

This example creates a variable that is like a variable in the master data dictionary.

```
set table
  name=TABABC ;
create variable
  name=NIORATE
  like='SUPPLIED._HPOV.NETIO' ;
```

Example 3

This example adds a supplied table definition to the active PDB and then creates a new variable in the table definition in the active PDB.

```
/*    NB:- Needs update access to the active PDB.       */
/*    A list of initial statistics is provided by       */
/*    the CREATE VARIABLE control statement. Because    */
/*    no SET DEFAULT control statement exists earlier   */
/*    in the  statement stream, any initial statistic   */
/*    not mentioned in the ISTATS= list on the CREATE   */
/*    VARIABLE control statement is set to its built-in */
/*    default, which is the NO state.                   */
  delete table
/* deletes table & data from PDB in case job is re-run  */
    name=tabxyz noerror ;
  add table
/*adds table from master data dictionary               */
    name=tabxyz ;
  create variable
/*creates new table in table in active PDB             */
    name=uuxx length=8 type=numeric format=5.
    interpret=count label='Special variable'
    description='Site-specific special variable'
    istats=(RANGE AVERAGE)
    ;
```

DELETE DERIVED

Deletes a derived variable definition from a table

DELETE DERIVED Overview

The DELETE DERIVED control statement deletes a derived variable definition from the active table in the active PDB.

Multiple parameters must be separated by at least one space.

DELETE DERIVED Syntax

DELETE DERIVED
NAME=*derived-var-name*
<**NOERROR**>;

Details

NAME=*derived-var-name*
is the name of the derived variable that you want to delete. The name must already exist in the dictionary.

NOERROR
causes error conditions to be treated as warnings for the current %CPDDUTL job. This is useful if some included statements refer to elements that do not exist and you still require subsequent statements to be processed.

By default, NOERROR is not specified; therefore, %CPDDUTL ends if an error occurs. If you specify the NOERROR parameter, %CPDDUTL does not end if an error occurs. Instead, a warning message is issued, and the job continues to run.

DELETE DERIVED Notes

The active table must be set before you can run the DELETE DERIVED control statement. To specify the active table, see “SET TABLE” on page 628.

If you want to stop using the derived variable but do not want to delete the definition and the data, then use the UPDATE DERIVED control statement with KEPT=NO. Data will no longer be processed or reduced for the variable, and the data will be deleted, but the definition will still exist.

Note: The DELETE DERIVED control statement deletes both the derived variable definition and the data. △

DELETE FORMULA

Deletes a formula variable definition from a table

DELETE FORMULA Overview

The DELETE FORMULA control statement deletes a formula variable definition from the active table in the active PDB.

Multiple parameters must be separated by at least one space.

DELETE FORMULA Syntax

DELETE FORMULA
    NAME=*formula-var-name*
    <**NOERROR**>;

Details

NAME=*formula-var-name*
is the name of the formula variable to be deleted. The name must already exist in the active PDB's data dictionary.

NOERROR
causes error conditions to be treated as warnings for the current %CPDDUTL job. This is useful if some included statements refer to elements that do not exist and you still require subsequent statements to be processed.

By default, NOERROR is not specified; therefore, %CPDDUTL ends if an error occurs. If you specify the NOERROR parameter, %CPDDUTL does not end if an error occurs. Instead, a warning message is issued, and the job continues to run.

DELETE FORMULA Notes

The active table must be set before you can run the DELETE FORMULA control statement. To specify a table as the active table, see "SET TABLE" on page 628.

If you want to stop calculating the formula variable but want to retain its definition in the table, then use the UPDATE FORMULA control statement with KEPT=NO.

Note: The DELETE FORMULA control statement deletes only the definition; it does not delete any data. △

DELETE TABLE

Deletes a table's data, variable definitions, and table definition from the active PDB

DELETE TABLE Overview

The DELETE TABLE control statement deletes the data from the table in the active PDB. It also deletes the definitions of all the variables in the table and deletes the definition of the table from the active PDB's data dictionary.

Note: Multiple parameters must be separated by at least one space. △

DELETE TABLE Syntax

DELETE TABLE
    NAME=*table-name* | _ALL_
    <**NOERROR**>;

Details

NAME=*table-name* | _ALL_
is the name of the table to be deleted. The table name must already exist in the data dictionary. Specify _ALL_ to delete all tables.

Note: If you specify _ALL_, then you will not be prompted to confirm this request. △

NOERROR
causes error conditions to be treated as warnings for the current %CPDDUTL job. This is useful if some included statements refer to elements that do not exist and you still require subsequent statements to be processed.

By default, NOERROR is not specified; therefore, %CPDDUTL ends if an error occurs. If you specify the NOERROR parameter, %CPDDUTL does not end if an error occurs. Instead, a warning message is issued, and the job continues to run.

DELETE TABLE Notes

The previous setting of the active table is not changed after you run the DELETE TABLE control statement, regardless of the success or failure of the DELETE TABLE control statement. However, if the table you delete is the active table, then the active table will be undefined when the DELETE TABLE control statement completes its execution. The active table must be redefined by using a SET TABLE control statement.

If you want to stop using the table but do not want to delete the definitions and data, do not use DELETE TABLE. Instead, use the UPDATE TABLE control statement with KEPT=NO. The table definition and the data will be ignored but not deleted.

If you want to delete the data in the table but keep the table definition, then use the PURGE TABLE control statement.

DELETE VARIABLE

Deletes a regular variable's definition and its data from a table

DELETE VARIABLE Overview

The DELETE VARIABLE control statement deletes a regular variable's definition from the active table in the active PDB. It also deletes the variable's data from the PDB and rebuilds the views when %CPDDUTL macro completes its execution.

Multiple parameters must be separated by at least one space.

DELETE VARIABLE Syntax

DELETE VARIABLE
NAME=*variable-name*
<**NOERROR**>;

Details

NAME=*variable-name*

is the name of the variable to be deleted. The name must already exist in the data dictionary.

NOERROR
causes error conditions to be treated as warnings for the current %CPDDUTL job. This is useful if some included statements refer to elements that do not exist and you still require subsequent statements to be processed.

By default, NOERROR is not specified; therefore, %CPDDUTL ends if an error occurs. If you specify the NOERROR parameter, %CPDDUTL does not end if an error occurs. Instead, a warning message is issued, and the job continues to run.

DELETE VARIABLE Notes

The active table must be set before you can run this statement. To specify a table as the active table, see "SET TABLE" on page 628.

If you want to stop using the variable but do not want to delete the definition, do not use DELETE VARIABLE. Instead, use the UPDATE VARIABLE control statement with KEPT=NO. The variable definition will be ignored but not deleted. The existing data will be deleted.

END

Ends the dictionary utility

END Overview

Although the dictionary utility (%CPDDUTL macro) stops reading statements as soon as the END control statement is encountered, normal end processing continues. That is, the syntax is not checked for control statements following the END control statement, and %CPDDUTL does not act on those control statements. Views are built for any tables modified by control statements preceding the END control statement, even if the BUILD VIEWS control statement is not specified within the %CPDDUTL control statements.

If you do not use an END, STOP, or QUIT control statement at the end of the %CPDDUTL control statement stream, then an END control statement is implied.

END Syntax

END;

END Example

The following example ends the dictionary utility. Views are rebuilt when the END control statement is encountered.

```
delete table
    name=TABXYZ noerror ;
add table
    name=TABXYZ ;
update variable
```

```
    name= ...
  .. other statements ..
end ;           /* end now and build views */
  .. other statements never executed ..
```

GENERATE SOURCE

Generates initial %CPDDUTL control statements for creating tables

GENERATE SOURCE Overview

The GENERATE SOURCE control statement enables you to generate %CPDDUTL control statements (such as CREATE TABLE, CREATE VARIABLE, CREATE FORMULA, CREATE DERIVED, and UPDATE TABLE) from the following sources:

- an existing SAS data set or view
- an existing table in a PDB (through its view)
- a file that has a recognized file format.

The %CPDDUTL control statements that are created by GENERATE SOURCE can be used to create or re-create a SAS IT Resource Management table. The generated control statements can be stored in an output file or a SAS catalog entry of type .SOURCE.

This control statement is also useful if you want to generate the control statements that you can later use to re-create your PDB or that another user can use to create a PDB table that is based on an existing table.

When you use a SAS data set or view (by specifying the DATASET= parameter) or a recognized file format (by specifying the INFILE= parameter) as input for GENERATE SOURCE, review and verify the accuracy of the generated statements before executing them. Specifically, review the DATETIME and DURATION variables, the interpretation types (see "Variable Interpretation Types" on page 697), the associated statistics (see "Statistic Codes for Variables" on page 706), and the default age limits for each level in the CREATE TABLE control statement that is created by GENERATE SOURCE. The GENERATE SOURCE control statement attempts to assign attributes such as interpretation type by using information such as words in the variable labels or format types in the SAS data set or view. However, these attributes may not always be suitable for your purposes. Check the generated control statements and use UPDATE control statements (UPDATE VARIABLE, UPDATE FORMULA, UPDATE DERIVED, and UPDATE TABLE) to update values as needed. Also, by using the UPDATE TABLE control statement, specify the BY variables list, CLASS variables lists, ID variables lists, and INDEX variables lists, because these cannot be read from the input data set.

For information about how the table and variable statements are constructed, see the topic "Appendix 8: Algorithm Used by GENERATE SOURCE" in the chapter "Setup: Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

Multiple parameters must be separated by at least one space.

GENERATE SOURCE Syntax

GENERATE SOURCE
DATASET=*'libref.name'*
ENTRYNAME=*'SAS-catalog-entry'*
FILENAME=*'physical-filename'*

INFILE=*'input-file-name'*
TABLE=<*input-table-name* | _ALL_>
<**DATETIME=***variable-name*>
<**DELIM=***'delimiter'*>
<**DURATION=***variable-name*>
<**INTRPRTDF=***valid-numeric-interpretation-type*>
<**INTYPE=***input-file-type*>
<**NAME=***new-table-name*>
<**REPLACE | NOREPLACE | APPEND**>
<**TYPE=**INTERVAL | EVENT>;

Details

DATASET=*'libref.name'*
names the SAS data set, SAS DATA step view, PROC SQL view, or SAS/ACCESS view for which you want to create dictionary control statements. Use the format of *libref.name* for the data set name. This data set must be defined in a SAS library; the libref that points to the library must be defined before this statement is encountered; and the data set name must be enclosed in quotation marks.

You must specify either this parameter, the INFILE= parameter, or the TABLE= parameter.

Note: Before you run the GENERATE SOURCE control statement against a data set, in the data set check that no label contains a single quotation mark (for example, *'s* as in *machine's*). You can check by submitting the following SAS code and looking at the results in the SAS output.

```
proc contents data=libref.data_set_name;
   run;
```

For more information about submitting SAS code, see the sample batch jobs in "Overview of SAS IT Resource Management Macros" on page 1 or see the section "Work with the Interface for Batch Mode" in "Chapter 2: Getting Started" in the *SAS IT Resource Management User's Guide*.

If you do find a single quotation mark and you have write access to the SAS DATA step code that created that data set, then remove the single quotation mark from the code and rerun the SAS DATA step. △

ENTRYNAME=*'SAS-catalog-entry'*
is the name of a SAS catalog entry. The name must be a four-level entry name and must be enclosed in single quotation marks.

CAUTION:
The entry type must be .SOURCE. △

The value of this parameter specifies one of the following:

- the location of the SAS catalog entry in which to save output from the SAVE SOURCE or GENERATE SOURCE control statement
- the location of the SAS catalog entry from which to obtain input for the INCLUDE SOURCE control statement.

You must specify either this parameter or the FILENAME= parameter, but not both.

FILENAME=*'physical-filename'*
is the physical location and name of a file, external z/OS data set, or PDS member. The value of this parameter specifies one of the following:

- the location in which to save the output from the SAVE SOURCE or GENERATE SOURCE control statement
- the location from which to include the input for the INCLUDE SOURCE control statement.

The specified location includes the complete directory path or high-level qualifiers plus the name of the file, external z/OS data set, or PDS member. *The name must be enclosed in single quotation marks. You must specify either this parameter or the ENTRYNAME= parameter, but not both.*

INFILE='*input-file-name*'
is the name of the input file for which you want %CPDDUTL control statements to be generated. *The filename must be enclosed in quotation marks.* The GENERATE SOURCE control statement reads the input file and creates control statements that you can then run to add tables to your PDB in order to support your data.

To execute the GENERATE SOURCE control statement, you must specify either this parameter, the DATASET= parameter, or the TABLE= parameter. If you want to generate control statements for DSI data (nonstandard HP OpenView Performance Agent), then you must specify the INFILE= parameter. If you specify the INFILE= parameter, then you must also specify the INTYPE= parameter and either the ENTRYNAME= parameter or the FILENAME= parameter.

The input filename must represent the complete physical location of the file (z/OS) or the complete directory pathname and filename (Windows or UNIX). For example, in Windows you would specify the complete directory path as shown in this example:

```
INFILE='e:\itrm\myfile.txt'
```

On UNIX you would specify the complete directory pathname and filename. On z/OS you would specify the fully qualified data set name as shown in this example:

```
INFILE='sasxyz.mydata.inputfile'
```

If you specify this parameter and the NAME= parameter, then the specified table name will be used for the first table that is created from the generated control statements. For each additional table that is created from these control statements, the specified table name will be altered to avoid duplicate table names. If you specify the INPUT= parameter and you specify TABLE=_ALL_, then the NAME= parameter is ignored.

Before you submit the generated control statements, verify that the statements are suitable for your purpose.

Note:

1. Before you can generate control statements for DSI data, the input file must be converted to the appropriate DSI format, by using HP OpenView Performance Agent utilities. For more information about this conversion process, see Section 1, Task 3, Action 1's information for UNIX or Windows in the *SAS IT Resource Management Server Setup Guide*.
2. A value of INTYPE=NTSMF requires that the INFILE= parameter refer to a text file that contains discovery records from an NTSMF log file. GENERATE SOURCE then uses the discovery records to construct table definitions.
3. A value of INTYPE=PATROL requires that the INFILE= parameter refer to a text file that has been produced by the dump_hist.exe program that is provided as part of the BMC Patrol software. GENERATE SOURCE will create table definitions for all objects present in the dumped history log file.

△

TABLE=*input-table-name* | _ALL_

is the name of the table for which the dictionary control statements should be created. This table must already exist in the active PDB. Specify _ALL_ to create dictionary control statements for all tables in the active PDB.

You must specify either this parameter, the INFILE= parameter, or the DATASET= parameter.

DATETIME=*variable-name*
specifies the name of a variable in the source data set that you want to use for the DATETIME variable in the generated table. This parameter is valid only when the DATASET= parameter is also specified. The DATETIME variable name is set in the generated source by using the NAME= parameter and the EXTNAME=*variable-name*.

This parameter is necessary only if a variable that is named DATETIME is absent from the input data.

DELIM='*delimiter*'
specifies the character that is used as the delimiter in your file of incoming data. For example, if your data uses a bar as the delimiter, then you would specify

```
DELIM='|'
```

This parameter is used *only* when INTYPE=CDC. Specifying this parameter when INTYPE=CDC is not required. However, if you specify INTYPE=CDC and you do not specify this parameter, then your GENERATE SOURCE statements may be incorrect.

DURATION=*variable-name*
specifies the name of a variable in the source data set that you want to use for the DURATION variable in the generated table. This parameter is valid only when the DATASET= parameter is also specified. The DURATION variable name is set in the generated source by using the NAME=DURATION parameter and the EXTNAME=*variable-name*.

This parameter is necessary only if the variable that is named DURATION is absent from the input data.

INTRPRTDF=*valid-numeric-interpretation-type*
indicates the default numeric interpretation type to use for numeric metrics for which %CPDDUTL is unable to determine an interpretation type.

The default is GAUGE. COUNT is the only other *valid-numeric-interpretation-type* that you are likely to use; however, any valid numeric interpretation type is accepted.

Note: This parameter does not alter the default interpretation type for character metrics. △

INTYPE=*input-file-type*
indicates the type of data for which control statements are being generated. Supported values are as follows:

CDC
represents character-delimited data. If you specify this parameter, then also specify the DELIM= parameter. For more information about character-delimited data, see "Appendix 13: Character-Delimited Support in GENERATE SOURCE" in the chapter "Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

DSI
represents non-standard HP OpenView Performance Agent data. *This parameter is required if you want to generate control statements for DSI data (HP OpenView Performance Agent Data Source Integration or non-standard*

HP OpenView Performance Agent data). For more about DSI data, see Section 1, Task 3, Action 1's information for UNIX or Windows in the *SAS IT Resource Management Server Setup Guide*.

NTSMF
represents Demand Technology NTSMF software.

PATROL
represents BMC Patrol Software.

You must specify this parameter if you specify the INFILE= parameter.

NAME=*new-table-name*
names the SAS IT Resource Management table to use in the NAME= parameter of the CREATE TABLE statement that is generated. When TABLE= is specified and NAME= is not specified, the *input-table-name* on the TABLE= parameter is used. When DATASET= is specified and NAME= is not specified, the name is formed by prefixing 'U' to the first six characters of the value that is specified by the DATASET= parameter.

When multiple table definitions are generated in one execution of GENERATE SOURCE, the value of the NAME= parameter is used on the first table name. To maintain unique table names, the remaining table names are created by replacing the last one or more characters in that table name with numbers.

If you specify INTYPE=CDC, then the default name for the table that is created by GENERATE SOURCE is UCDCTAB.

REPLACE | NOREPLACE | APPEND
indicates whether the generated source should replace, leave alone, or be appended to the existing catalog or file.

REPLACE indicates that any existing source in either a catalog or a file will be replaced; NOREPLACE indicates that the existing source is not replaced; APPEND indicates that any generated source should be appended to the existing catalog or file.

The default is NOREPLACE.

TYPE=INTERVAL | EVENT
Sets the type of the table whose definition is being generated. TYPE= is valid only when the DATASET= parameter is specified or when INTYPE=CDC is specified.

If TYPE= is not specified and DATASET= is specified, then the default for TYPE= is EVENT. Otherwise, the default is INTERVAL.

GENERATE SOURCE Notes

After you edit the output from GENERATE SOURCE, you can use the edited output as input for the %CPDDUTL macro.

When using the GENERATE SOURCE control statement, the following items apply:

- DATETIME and LSTPDATE variables are automatically created.
- If you specify the table type as INTERVAL, then the DURATION variable is automatically created.
- Default statistics correlate with the interpretation type. Therefore, if you change the interpretation type on a variable before you run the output to create a table, then you might also need to change the statistics.
- For each variable, the initial statistics are determined according to the interpretation type. For more information on interpretation types, see the "Variable Interpretation Type Summary Table" on page 710 and "Variable Interpretation Types" on page 697.

Note: If you use the DATASET= parameter, consider that variable names longer than seven characters in the SAS data set or view are translated to seven-character names by truncating the original names and appending a number if there is a conflict. △

If you need to revise the draft of the control statements you are using to generate the source for your table and variable definitions, it is recommended that you use the UPDATE TABLE and UPDATE VARIABLE control statements, instead of editing the draft directly. (If you edit the control statements directly and then make changes to the staging code and regenerate the draft, you need to re-enter all the edits.) Follow this order when submitting the control statements:

```
%cpcat;
cards4;
<--- unedited draft of control statements
<--- edits for the draft
<--- customizations for the table and variables
;;;;
%cpcat (cat=work.temp.addcust2.source);
```

GENERATE SOURCE Examples

Example 1

This example generates %CPDDUTL control statements that define a table called TABXYZ, based on an existing table in the active PDB. The generated statements are written to a catalog entry, and any previous output in the catalog entry is replaced.

```
generate source
   table=TABXYZ                    /* existing table as input */
   entryname='SASUSER.DICT.TABXYZ.SOURCE'   /* output */
   replace ;
```

Example 2

This example generates %CPDDUTL control statements for all tables in the active PDB. The generated statements are written to an external file, and any previous output in the catalog entry is replaced.

```
generate source
   table=_ALL_       /* all existing tables in PDB as input */
   filename='physical-filename-source'   /* output */
   replace ;
```

Example 3

This example generates %CPDDUTL control statements that can be used to define the SAS data set SASUSER.TABABC as a table that is called TABLE1.

```
generate source
   name=TABABC                      /* new name to use */
   dataset='SASUSER.TABLE1'         /* SAS data set as input */
   entryname='SASUSER.DICT.TABLE1.SOURCE'    /* output */
   replace ;
```

Example 4

The following code will read the file from INFILE= and generate table and variable definitions which it will save in a SAS catalog entry. (This could be a text file if the FILENAME= parameter was used.)

```
%cpcat;
cards4;
generate source infile='location.of.file.with.metadata'
               intype=NTSMF
               entryname='work.ddutl.gensrc.source';
;;;;
%cpcat(cat=work.cpddutl.gen.source);
%cpddutl(entrynam=work.cpddutl.gen.source);
```

INCLUDE SOURCE

Includes additional control statements

INCLUDE SOURCE Overview

The INCLUDE SOURCE control statement enables you to include additional %CPDDUTL control statements into a stream of %CPDDUTL control statements. You can manage groups of control statements that perform a specific function (for example, delete and re-create a particular table) and then include and apply them with the %CPDDUTL control statements in order to perform larger tasks in one execution of the %CPDDUTL macro.

The control statements can be read from an external file or a SAS catalog entry. The same control statements can be included in the %CPDDUTL control statement stream in as many places as required.

Note: Do not include a group of control statements into itself. This recursion is not detected or prevented. △

INCLUDE SOURCE Syntax

INCLUDE SOURCE
ENTRYNAME=*'SAS-catalog-entry'*
FILENAME=*'physical-filename'*
<**NOERROR**>;

Details

ENTRYNAME=*'SAS-catalog-entry'*
is the name of a SAS catalog entry. The name must be a four-level entry name and must be enclosed in single quotation marks.

CAUTION:
The entry type must be .SOURCE. △

The value of this parameter specifies one of the following:

- the location of the SAS catalog entry in which to save output from the SAVE SOURCE or GENERATE SOURCE control statement

- the location of the SAS catalog entry from which to obtain input for the INCLUDE SOURCE control statement.

You must specify either this parameter or the FILENAME= parameter, but not both.

FILENAME=*'physical-filename'*
is the physical location and name of a file, external z/OS data set, or PDS member. The value of this parameter specifies one of the following:

- the location in which to save the output from the SAVE SOURCE or GENERATE SOURCE control statement
- the location from which to include the input for the INCLUDE SOURCE control statement.

The specified location includes the complete directory path or high-level qualifiers plus the name of the file, external z/OS data set, or PDS member. *The name must be enclosed in single quotation marks. You must specify either this parameter or the ENTRYNAME= parameter, but not both.*

NOERROR
causes error conditions to be treated as warnings for the current %CPDDUTL job. This is useful if some included statements refer to elements that do not exist and you still require subsequent statements to be processed.

By default, NOERROR is not specified; therefore, %CPDDUTL ends if an error occurs. If you specify the NOERROR parameter, %CPDDUTL does not end if an error occurs. Instead, a warning message is issued, and the job continues to run.

INSTALL TABLE

Copies the dictionary information for a PDB table into the master data dictionary

INSTALL TABLE Overview

The INSTALL TABLE control statement copies the dictionary information for a PDB table into the master data dictionary, replacing any information in the master data dictionary for the specified table. By installing your own table in the master data dictionary, you can use the table just as you would use a SAS IT Resource Management supplied table, and it can be used by other users at your site. The PDB is not changed by the INSTALL TABLE control statement. If you choose the REPLACE option, then the table is created or replaces an existing table definition that has the same name.

Multiple parameters must be separated by at least one space.

INSTALL TABLE Syntax

INSTALL TABLE
NAME=*table-name*
<**REPLACE | NOREPLACE**>
<**RETAIN | NORETAIN**>;

Details

NAME=*table-name*

is the name of the table definition to be installed. This parameter is required, and the specified table definition must exist in the active PDB. If this table definition name already exists in the master data dictionary, then you must specify the REPLACE parameter in order to install the new table definition.

REPLACE | NOREPLACE
specifies whether or not to replace an existing table definition in the master data dictionary, when the existing table definition and the new table definition have the same table name. REPLACE means that an existing table definition in the master data dictionary will be replaced with the table definition that is named in this statement. NOREPLACE means that an existing table definition in the master data dictionary will not be replaced. *The default is NOREPLACE*.

RETAIN | NORETAIN
specifies whether or not this table definition should be installed in the SITELIB library. RETAIN means that you want this table definition to be installed. NORETAIN means that you do not want this table definition to be installed. *The default is NORETAIN*.

When you specify RETAIN, you are updating the SITELIB library and therefore must have write access to SITELIB. To obtain write access to SITELIB, specify SITEACC=WRITE (UNIX or Windows) or SITEACC=OLD (z/OS) on the %CPSTART macro before using the %CPDDUTL macro and the INSTALL TABLE control statement.

INSTALL TABLE Notes

When the INSTALL TABLE control statement completes its execution, the active table is set to the table that is named in this statement. To specify a different table as the active table, see “SET TABLE” on page 628.

To run INSTALL TABLE, you must have a PGMLIBW libref. *(The PGMLIBW libname must be allocated prior to running %CPSTART.)* The libref must have write access to the same library that the PGMLIB libref points to. To see where PGMLIB points, issue the LIBNAME command from the command line. You can then issue the libname statement to provide write access. *On z/OS*, if PGMLIB points to 'YOUR.LIB.PGMLIB', for example, then issue the following command:

```
libname pgmlibw 'your.lib.pgmlib'
disp=old;
```

On UNIX, if PGMLIB points to /usr/local/sas/saspgm/dirname/pgmlib, for example, then issue the following command:

```
libname pgmlibw
'/usr/local/sas/saspgm/dirname/pgmlib';
```

In Windows, if PGMLIB points to c:\sas\cpe\pgmlib, for example, then issue the following command:

```
libname pgmlibw 'c:\sas\cpe\pgmlib' ;
```

This gives the INSTALL TABLE command update access to the master data dictionary, which is stored in PGMLIB. The INSTALL TABLE control statement will fail if any other user is using the SAS IT Resource Management application and has the PGMLIB library allocated.

When upgrading to a new version of SAS IT Resource Management, you can use the %CPSITEUP macro to integrate table definitions from your SITELIB into the new master data dictionary for SAS IT Resource Management. For more details, see the

appendix about site libraries in the SAS IT Resource Management installation instructions.

Update access to the root location where the SAS IT Resource Management application is installed must be available and allowed by your security system. If update access is not available, then an error message is issued and the table is not installed.

INSTALL TABLE Example

This example installs a table definition (TABXYZ). If a table definition with this name already exists in the master data dictionary, then it replaces the existing table definition with the new table definition. Also, because the RETAIN option is used, the table definition will be maintained when future updates are installed.

```
install table
     name=TABXYZ replace retain;
```

LIST DERIVEDS

Prints a list of derived variables

LIST DERIVEDS Overview

The LIST DERIVEDS control statement prints in the SAS log a list of all the derived variables and their labels in the active table in the active PDB.

LIST DERIVEDS Syntax

LIST DERIVEDS;

LIST DERIVEDS Notes

The active table must be set before you can run this statement. To specify a table as the active table, see “SET TABLE” on page 628.

To list the source statements that describe the calculations that are used for the derived variables, see “PRINT TABLE” on page 618.

LIST FORMULAS

Prints a list of formula variables

LIST FORMULAS Overview

The LIST FORMULAS control statement prints in the SAS log a list of all the formula variables and their labels in the active table in the active PDB.

LIST FORMULAS Syntax

LIST FORMULAS;

LIST FORMULAS Notes

The active table must be set before you can run this statement. To specify a table as the active table, see “SET TABLE” on page 628.

To list the source statements that describe the calculations used for the formula variables, see “PRINT TABLE” on page 618.

LIST TABLES

Prints a list of tables in the active PDB

LIST TABLES Overview

The LIST TABLES control statement prints a list of all the tables and their labels for the active PDB. The list is printed in the SAS log. The list can be useful at the end of a job that creates many tables, because the list confirms which tables are in the PDB.

LIST TABLES Syntax

LIST TABLES;

LIST TABLES Examples

Below is an example of what you might see in the SAS log after running the LIST TABLES control statement.

```
2
%cpddutl(entrynam=sasuser.cpddutl.list.source,list=y);
NOTE: Search path replaced, now includes CPE libname.
NOTE: Search path replaced, now includes CPE libname.
NOTE: CPDDUTL Started at 26FEB03:17:00:18 Active PDB is /u/username/newpdb/
NOTE: Source SASUSER.CPDDUTL.LIST.SOURCE contains 1 records.
00001  list tables;
NOTE: List of tables in PDB /u/username/newpdb/ follows:

Table    Label
-------------------------------------------------
F42      The test table TABXYZ
FF2      Default label 1
XDCCAPT  DCOLLECT tape capacity
XDCVOLS  DCOLLECT active volume information
XTY70    RMF CPU Activity and Address Space Stati
XTY72    RMF Workload Activity

NOTE: Processing of dictionary utility statements completed successfully.
NOTE: CPDDUTL Ended at 26FEB03:17:00:20 Elapsed time   0:00:01.99
```

LIST VARIABLES

Prints a list of regular variables in the active table

LIST VARIABLES Overview

The LIST VARIABLES control statement prints in the SAS log a list of all the regular variables and their labels in the active table in the active PDB. This is a good method of verifying the regular variables that exist in the table.

LIST VARIABLES Syntax

LIST VARIABLES;

LIST VARIABLES Notes

The active table must be set before you can run this control statement. To specify a table as the active table, see "SET TABLE" on page 628.

Formula and derived variables are not printed by this control statement. Use the LIST FORMULAS and LIST DERIVEDS control statements to print those variables.

MAINTAIN TABLE

Updates or replaces dictionary information in the active PDB with information from the master data dictionary

MAINTAIN TABLE Overview

This statement updates the table and variable definitions for supplied tables that are defined in a PDB. The active PDB's data dictionary is updated with the information in the master data dictionary for this table definition. This statement is useful for applying changes to a table when you install a maintenance release for the SAS IT Resource Management application.

Note: Multiple parameters must be separated by at least one space. △

MAINTAIN TABLE Syntax

MAINTAIN TABLE
NAME=*table-name* | _ALL_
<**REPORT | NOREPORT**>
<**SCOPE=**EXISTING | NEWVARS>
<**UPDATE | NOUPDATE**>;

Details

NAME=*table-name* | _ALL_

specifies the name of the table to be maintained. The table must exist in the active PDB and in the master data dictionary. Specify _ALL_ in order to maintain all tables in the active PDB.

REPORT | NOREPORT
specifies whether or not a report is to be generated in the SAS log. REPORT requests the report. NOREPORT indicates that the report is not needed. The statement will be faster if NOREPORT is used. *The default is REPORT*.

SCOPE=EXISTING | NEWVARS
specifies which parts of the PDB's table definition are replaced.

EXISTING
A limited number of updates are performed to existing definitions within the PDB, as follows:

1. The table label, description, and version will be updated.
2. The variable label, description, format, and informat will be updated.
3. If the length of a variable has been increased in the supplied data dictionary, then the length of the variable will be updated in the PDB.

NEWVARS
Only new variables will be added to the table. New variables are variables that exist in the master data dictionary's table definition but do not currently exist in the PDB's table definition.

The default is SCOPE=NEWVARS.

UPDATE | NOUPDATE
enables you to review the effects of the MAINTAIN TABLES control statement before you commit any changes to the PDB. NOUPDATE enables the review and does not commit any changes. UPDATE commits the changes. *The default is NOUPDATE*.

MAINTAIN TABLE Notes

The PDB should be backed up before you use the MAINTAIN TABLE control statement to apply maintenance. This statement may cause the data in the table to change if any of the following conditions occur.

- A variable is no longer available from the collector. Future observations will have missing values for these variables.
- SAS output formats are changed. This format change may alter the way the data is displayed.

This statement may take a long time to run, depending on the size of the table (number of observations). If NOUPDATE is specified with NOREPORT, then only the syntax and validity of the operands are tested and appropriate messages are written in the SAS log. However, this method can still take a long time.

Variables or formulas that you have added will not be removed. However, if there is a newly supplied variable with the same name as an existing variable, then the attributes of the new variable will replace the attributes of the existing variable. Variable-name conflicts of this type may be detected by using the REPORT and NOUPDATE parameters. The only way to resolve this type of conflict is to delete or rename the user-defined variable or formula before applying the maintenance.

MAINTAIN TABLE Example

Example 1

This example adds new variables from the master data dictionary to the table definition of the TABXYZ table. A report is not printed in the SAS log.

```
maintain table
     name=TABXYZ update noreport ;
```

Example 2

This example updates all tables in the PDB. The attributes of the table definition are maintained, new variables are added, and existing variables are modified. The report of the changes is not printed in the SAS log.

```
maintain table
      name=_ALL_ scope=existing
      update noreport ;
```

If you want to update the PDB and add only new variables (without modifying existing variables), then specify the following.

```
maintain table
     name=_all_
     report
     scope=newvars
     update;
```

Example 3

This example uses the NOUPDATE parameter to report on changes that would occur if you used the UPDATE parameter. NOUPDATE enables you to review all possible changes before applying the updates to the dictionary.

```
maintain table
    name= _ALL_ scope=existing
    noupdate report  ;
```

PRINT SOURCE

Prints messages during execution of %CPDDUTL

PRINT SOURCE Overview

The PRINT SOURCE control statement prints in the SAS log a message that you specify.

PRINT SOURCE Syntax

PRINT SOURCE
SOURCE={

any ASCII text
};

Details

SOURCE={*multiple source lines*}
is the ASCII text that you want to save. Place the text within the braces and be sure to place the braces on separate lines as shown in the following example.

```
save source
     source={
             +-------+
             |       |
             | TEST  |
             |       |
             +-------+
            }
     entryname='sasuser.save.mysource' ;
```

Note: If you place the braces in other locations, such as on the same line as the text, then your output might not appear in the same format that you used. △

PRINT TABLE

Prints a table definition from the active PDB

PRINT TABLE Overview

The PRINT TABLE control statement prints a table definition from the active PDB. This includes all the table's variable, derived, and formula variable definitions and the source statements for the formula variables. The output is printed in the SAS log.

PRINT TABLE Syntax

PRINT TABLE
NAME=*table-name* | _ALL_ ;

Details

NAME=*table-name* | _ALL_
is the name of the table definition to be printed. The name must already exist in the active PDB's data dictionary. Specify _ALL_ in order to print all table definitions in the active PDB.

PRINT TABLE Notes

When the PRINT TABLE control statement successfully completes its execution, the active table is set to the table that is named in this statement (if a single table is

printed) or to the last table that is printed (if all tables are printed). To specify a different table as the active table, see "SET TABLE" on page 628.

If you specify NAME=_ALL_ on this PRINT TABLE control statement and subsequent statements depend on the value of the active table, then you can use a SET TABLE control statement to explicitly set the value of the active table.

The PRINT TABLE control statement generates output that is similar to (but not exactly the same as) the set of control statements that are used to create the table; however, the output cannot be used as input later. If you want to generate output that can later be used as input, then use the GENERATE SOURCE control statement instead.

Note: Use the PRINT TABLE control statement with caution, because it produces a large amount of output. △

PURGE TABLE

Deletes the data in a table in the active PDB without deleting the table's definition

PURGE TABLE Overview

The PURGE TABLE control statement purges data from a table in the active PDB. The data dictionary definitions of the table, its variables, and its formula variables are unchanged. The result is an empty table.

PURGE TABLE Syntax

PURGE TABLE
    NAME=*table-name* | _ALL_ ;

Details

NAME=*table-name* | _ALL_
is the name of the table to be purged. The name must already exist in the active PDB's data dictionary. Specify a specific table name or specify _ALL_ in order to purge all tables.

PURGE TABLE Notes

When this control statement successfully completes its execution, the active table is set to the table that is named in this statement (if a single table is purged) or to the last table that is purged (if all tables are purged). If you specify NAME=_ALL_ and you depend on the value of the active table for statements that follow this statement, then use the SET TABLE control statement after this statement to explicitly set the value of the active table.

To delete the data in a table and also to delete the table's definition and the definitions of all its variables, use the DELETE TABLE control statement.

QUIT

Stops the dictionary utility

QUIT Overview

The QUIT control statement enables you to stop the %CPDDUTL dictionary utility upon reaching this statement. This statement is an alias of the STOP control statement. As with the STOP control statement, no views are rebuilt.

If you do not code an END, STOP, or QUIT control statement at the end of the %CPDDUTL control statement stream, then an END control statement is implied.

QUIT Syntax

QUIT;

SAVE SOURCE

Saves ASCII text (that you supply) into an external file or a SAS catalog entry

SAVE SOURCE Overview

By using the SAVE SOURCE control statement, you can save control statements (that you supply) into an external file or a SAS catalog entry.

Multiple parameters must be separated by at least one space.

SAVE SOURCE Syntax

SAVE SOURCE
ENTRYNAME=*'SAS-catalog-entry'*
FILENAME=*'physical-filename'*
SOURCE={
any single or multiple lines of ASCII text
};

Details

ENTRYNAME=*'SAS-catalog-entry'*
is the name of a SAS catalog entry. The name must be a four-level entry name and must be enclosed in single quotation marks.

CAUTION:
The entry type must be .SOURCE. △

The value of this parameter specifies one of the following:

- the location of the SAS catalog entry in which to save output from the SAVE SOURCE or GENERATE SOURCE control statement

- the location of the SAS catalog entry from which to obtain input for the INCLUDE SOURCE control statement.

You must specify either this parameter or the FILENAME= parameter, but not both.

FILENAME=*'physical-filename'*
is the physical location and name of a file, external z/OS data set, or PDS member. The value of this parameter specifies one of the following:

- the location in which to save the output from the SAVE SOURCE or GENERATE SOURCE control statement
- the location from which to include the input for the INCLUDE SOURCE control statement.

The specified location includes the complete directory path or high-level qualifiers plus the name of the file, external z/OS data set, or PDS member. *The name must be enclosed in single quotation marks. You must specify either this parameter or the ENTRYNAME= parameter, but not both.*

SOURCE={*multiple source lines*}
is the ASCII text that you want to save. Place the text within the braces and be sure to place the braces on separate lines as shown in the following example.

```
save source
     source={
             +-------+
             |       |
             | TEST  |
             |       |
             +-------+
            }
     entryname='sasuser.save.mysource' ;
```

Note: If you place the braces in other locations, such as on the same line as the text, then your output might not appear in the same format that you used. △

SAVE SOURCE Notes

You can use the %CPCAT macro as an alternate way to copy statements into a source catalog entry.

SET DEFAULTS

Sets defaults (which are used during the current call to %CPDDUTL) for creating table and variable definitions

SET DEFAULTS Overview

You can use the SET DEFAULTS control statement to specify default values for many of the parameters on the CREATE TABLE, CREATE VARIABLE, CREATE DERIVED, and CREATE FORMULA statements. The defaults that you specify with

the SET DEFAULTS control statement expire at the end of the current invocation of the %CPDDUTL macro.

If you do not specify values for all parameters on a CREATE TABLE, CREATE VARIABLE, CREATE DERIVED, or CREATE FORMULA control statement, then the unspecified parameters obtain their defaults from the values that are specified by a SET DEFAULTS statement earlier in the control statement stream. If you do not specify a parameter on a CREATE statement and did not specify a value for that parameter on a SET DEFAULTS statement earlier in the control statement stream, then the default values from the built-in defaults table are used (see "Built-in Defaults for the SET DEFAULTS Statement" on page 627).

Note: You must explicitly specify values for the parameters NAME=, BYVARS=, CLASSVARS=, IDVARS=, and INDEXVARS=. They have no default values. △

If you issue a second SET DEFAULTS statement during the current %CPDDUTL statement stream, then the first set of defaults are updated by the second set of defaults. For example, if you specify

```
SET DEFAULTS COLLECTOR=GENERIC
LABEL='LABEL1'
```

and then specify

```
SET DEFAULTS COLLECTOR=SPECTRUM
```

then the original value for LABEL= and the new value for COLLECTOR= become the defaults.

Multiple parameters must be separated by at least one space.

SET DEFAULTS Syntax

SET DEFAULTS
<**COLLECTOR=***collector* | generic>
<**DESCRIPTION=**'*string*'>
<**DETAIL=**(<AGELIMIT=*number*>)>
<**DAY=**(<AGELIMIT=*number*><*statistic-keywords-list*>)>
<**WEEK=**(<AGELIMIT=*number*><*statistic-keywords-list*>)>
<**MONTH=**(<AGELIMIT=*number*><*statistic-keywords-list*>)>
<**YEAR=**(<AGELIMIT=*number*><*statistic-keywords-list*>)>
<**EXTNAME=***name*>
<**FORMAT=***SAS-format*>
<**IDNUM=***number*>
<**INFORMAT=**>
<**INTERPRET=***interpretation*>
<**ISTATS=**(*statistic-keywords-list*)>
<**KEPT=**YES | NO>
<**LABEL=**'*string*'>
<**LENGTH=***number*>
<**LEVELS=**'<DETAIL><DAY><WEEK><MONTH><YEAR>'>
<**OID=***oid-specification*>
<**SUBJECT=**'*subject-keyword-list*'>
<**TOOLNAME=***name*>
<**VALIDITY=**>;

Details

COLLECTOR=*collector-name* | *generic*

is the name that (along with the tool name) enables the process task to access and run the correct staging code and use the correct table and variable definitions. The value of the collector name specifies the name of the data collector or data source that creates the log file data. (Which data collectors or data sources are supported by a host depends on the SAS IT Resource Management software license. For more information, see the SAS IT Resource Management installation instructions.) This is the value that you specify for the COLLECTR= parameter on the process macro (%CxPROCES) when you process your data into the PDB.

Note: %CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES. △

For more about the value to use for *collector-name*, see the Notes section for the UPDATE TABLE control statement.

If the name of the collector contains special characters, then enclose the name in single quotation marks. Here is an example, if you are using HP-OVPA:

```
collector='hp-ovpa'
```

DESCRIPTION='*string*'
specifies a default description string for tables, regular variables, derived variables, and formula variables. See the following %CPDDUTL control statements for more information: "CREATE TABLE" on page 587, "CREATE VARIABLE" on page 591, "CREATE DERIVED" on page 577, and "CREATE FORMULA" on page 584.

DETAIL=(AGELIMIT=*number*)
DAY=(<AGELIMIT=*number*><*statistic-keywords-list*>)
WEEK=(<AGELIMIT=*number*><*statistic-keywords-list*>)
MONTH=(<AGELIMIT=*number*><*statistic-keywords-list*>)
YEAR=(<AGELIMIT=*number*><*statistic-keywords-list*>)
AGELIMIT= defines the time span for data to be retained at each level. For detail and day levels, the number represents days. For week level, the number represents weeks; for month level, it represents months; and for year level, it represents years. The *number* must be an integer.

Note: If AGELIMIT=0 at detail level, then your data is stored in the detail level until the next reduce or process step runs. Therefore, if the reduce step runs immediately after the process step runs, then the data is processed, reduced into the summary levels, and removed from the detail level. However, if the process step runs again before the data has been reduced, then the existing data in the data level is removed before the summary levels are updated. Using AGELIMIT=0 at the detail level is useful if you are collecting and processing large amounts of data on a daily basis; however, to avoid losing data at summary levels, always reduce the data immediately after you process it.

AGELIMIT=0 at any level other than detail level indicates that data is not to be stored at that level. △

Note: For information on setting age limits for your data, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*. △

The *statistic-keywords-list* is the list of statistics required for this variable at a specific reduction level. The list can contain one or more of the following statistics:

AVERAGE | NOAVERAGE
for the arithmetic mean

COUNT | NOCOUNT
for an accumulated count of the number of observations that have nonmissing values for the variable, starting at 0

CV | NOCV

for the coefficient of variation

MAXIMUM | NOMAXIMUM
for the maximum value

MINIMUM | NOMINIMUM
for the minimum value

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

For more information about the age limit, see the %CPDDUTL control statement: "CREATE TABLE" on page 587. For more information about the statistics, see the %CPDDUTL control statements: "CREATE VARIABLE" on page 591, and "CREATE DERIVED" on page 577.

EXTNAME=*name*
specifies a default external name for tables and variables. See the following %CPDDUTL control statements: "CREATE TABLE" on page 587 and "CREATE VARIABLE" on page 591.

FORMAT=*SAS-format*
specifies the default SAS format to use. See the following %CPDDUTL control statements: "CREATE VARIABLE" on page 591, "CREATE DERIVED" on page 577, and "CREATE FORMULA" on page 584.

IDNUM=*number*
specifies a numeric identifier that has different uses for each collector. Refer to the information about a specific collector in the *SAS IT Resource Management Server Setup Guide*.

This parameter is used only for UNIX and Windows.

INFORMAT=
is not currently used.

INTERPRET=*interpretation*
indicates how to calculate statistics for the regular or derived variable and how to use the variable in the reduce task and build views task. See the %CPDDUTL control statements "CREATE VARIABLE" on page 591 and "CREATE DERIVED" on page 577.

ISTATS=(*statistic-keywords-list*)
lists the default statistics for a variable or derived variable. The statistics are saved with the variable definition, but they are not calculated within the active

table. These statistics are used when you run INSTALL TABLE or ADD TABLE control statements, because the master data dictionary does not keep reduction level statistics information.

When called by the UPDATE VARIABLE control statement or the UPDATE DERIVED control statement, ISTATS= applies changes to the default statistics. If an initial statistic is not specified using the UPDATE VARIABLE control statement, the existing setting for that statistic is not changed.

When called by the SET DEFAULTS control statement, ISTATS= sets the default statistics for a variable if you do not set the default statistics when you create the variable.

You can specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request at each level are calculated for each class variable group in each summary level.

The statistics that are calculated for the active table are those that are specified by the DAY=, WEEK=, MONTH=, and YEAR= parameters. Therefore, when you install a table in the master data dictionary, it stores the ISTATS with the table definition. When you later add that table to another PDB, the ADD TABLE statement populates the DAY=, WEEK=, MONTH=, and YEAR= statistics based on the ISTATS= list of statistics. You can then customize the added table by using the SAS IT Resource Management GUI or an UPDATE VARIABLE/UPDATE DERIVED statement with the DAY=, WEEK=, MONTH=, and YEAR= parameters.

AVERAGE | NOAVERAGE
: for the arithmetic mean for all observations

COUNT | NOCOUNT
: for an accumulated count of the number of observations that have nonmissing values, starting at zero

CV | NOCV
: for the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAXIMUM | NOMAXIMUM
: for the maximum value for all observations

MINIMUM | NOMINIMUM
: for the minimum value for all observations

NMISS | NONMISS
: for the number of observations with missing variable values

RANGE | NORANGE
: for the difference between MAXIMUM and MINIMUM

STD | NOSTD
: for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
: for the sum of values for all observations

USS | NOUSS
: for the uncorrected sum of squares

VARIANCE | NOVARIANCE

for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: ISTATS= can be specified on the SET DEFAULTS, CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, and UPDATE DERIVED statements. If you install or add a table for which ISTATS have not been specified, then the table is given the default statistic values for ISTATS=, which have been specified in SET DEFAULTS for the current set of %CPDDUTL statements. If default values have not been set via SET DEFAULTS, then the variable is treated as if all the statistics in the ISTATS= list were set to NO. If the table is installed and then added, then the added table does not keep any statistics at the day, week, month, or year level unless you set them by using an UPDATE statement. △

If you use a SET DEFAULTS statement for a single variable, then a useful default setting is one that is based on the variable's interpretation. (For more about the relation between interpretation and initial statistics, see "GENERATE SOURCE" on page 604.)

If the SET DEFAULTS statement will apply to a number of variables, then a useful default setting is ISTATS=(AVERAGE).

KEPT=YES | NO
specifies a default KEPT status. See the following %CPDDUTL control statements: "CREATE TABLE" on page 587, "CREATE VARIABLE" on page 591, "CREATE DERIVED" on page 577, and "CREATE FORMULA" on page 584.

LABEL='*string*'
specifies a default label for tables, variables, and formula variables. See the following %CPDDUTL control statements: "CREATE TABLE" on page 587, "CREATE VARIABLE" on page 591, "CREATE DERIVED" on page 577, and "CREATE FORMULA" on page 584.

LENGTH=*number*
specifies the default length to use. See the following %CPDDUTL control statements: "CREATE VARIABLE" on page 591, "CREATE DERIVED" on page 577, and "CREATE FORMULA" on page 584.

LEVELS='*levels*'
specifies the levels (DETAIL, DAY, WEEK, MONTH, YEAR) at which a formula variable is to be available. If the formula variable relies on other variables, then those variables must exist at all the levels that are specified. For more information, see the %CPDDUTL control statement "CREATE FORMULA" on page 584.

OID=*oid-specification*
is an SNMP Object ID of the form n.n.n.n.n.n. The value must be unique for the variable if you specify it. Use *OID=.* to specify none. For more information, see the %CPDDUTL control statement "CREATE VARIABLE" on page 591.

This parameter is used only on UNIX.

SUBJECT='*subject-keyword-list*'
specifies the default subject list to use. See the following %CPDDUTL control statements: "CREATE VARIABLE" on page 591, "CREATE DERIVED" on page 577, and "CREATE FORMULA" on page 584.

TOOLNAME=*name*
is the name that (along with the collector-name) enables the process task to access and run the correct staging code and to use the correct table and variable definitions.

For more about the value to use for tool-name, see the Notes section for the UPDATE TABLE control statement.

This name cannot exceed 32 characters.

VALIDITY=
is not currently used.

SET DEFAULTS Notes

If the value for a parameter is not specified in a CREATE TABLE, CREATE VARIABLE, CREATE DERIVED, or CREATE FORMULA statement and the value for that parameter is not specified in the SET DEFAULTS statement, the default values are used (see "Built-in Defaults for the SET DEFAULTS Statement" on page 627).

Parameter defaults are used only on the CREATE statements, not on the UPDATE statements.

SET DEFAULTS Example

This example adds a new regular variable to a table.

```
 /*    - Needs update access to the active PDB.              */
 /*    - This control statement stream                      */
 /*        sets the default initial statistics to ON        */
 /*          for RANGE and AVERAGE; the other               */
 /*          default initial statistics remain OFF          */
 /*        creates a new variable in the table in the       */
 /*           active PDB                                    */
 /*    - The CREATE VARIABLE statement does not provide     */
 /*      its own settings of initial statistics, so for     */
 /*      istats= the default from SET DEFAULTS is used.     */
 /*      For parameters like kept=, the built-in            */
 /*      defaults are used.       */
set table
    name=utab;

 set defaults
   istats= ( RANGE AVERAGE )
   ;
 create variable
   name=uuxx length=8 type=numeric format=5.
   interpret=count label='Special variable'
   description='Site-specific special variable' ;
```

Built-in Defaults for the SET DEFAULTS Statement

Table 4.3 Built-in Defaults for the SET DEFAULTS Control Statement

Attribute	Built-in Default
COLLECTOR=collector	blank
DAY=(AGELIMIT=number)	45
DAY=(CLASSVARS='name-list')	none

Attribute	Built-in Default
DAY=(statistic-keywords)	none
DESCRIPTION='string'	blank
DETAIL=(AGELIMIT=number)	10
DETAIL=(BYVARS='name-list')	none
EXTNAME=name	blank
FORMAT=SAS-format	blank
INTERPRET=interpretation	blank
ISTATS=	blank
KEPT=YES | NO	NO
LABEL='string'	blank
LENGTH=number	0
LEVELS='reduction-level-list'	none
MONTH=(AGELIMIT=number)	18
MONTH=(CLASSVARS='name-list')	none
MONTH=(statistic-keywords-list)	none
NAME=name	blank
SUBJECT='subject-keyword-list'	none
TOOLNAME=name	blank
WEEK=(AGELIMIT=number)	15
WEEK=(CLASSVARS='name-list')	none
WEEK=(statistic-keywords-list)	none
YEAR=(CLASSVARS='name-list')	none
YEAR=(AGELIMIT=number)	5
YEAR=(statistic-keywords-list)	none

SET TABLE

Sets the name of the active table

SET TABLE Overview

The SET TABLE control statement specifies the value of active table. The active table is the table to which your actions will apply. For example, control statements that perform actions on regular variables, derived variables, and formula variables apply to those variables in the active table.

When you use the SET TABLE control statement to set the active table, all subsequent control statements are applied to the table that is named in this statement until you reset the active table or until a control statement resets the active table.

SET TABLE Syntax

SET TABLE
NAME=*table-name*;

Details

NAME=*table-name*
is the name of the table to use as the current table. The name must already exist in the data dictionary of the active PDB.

SET TABLE Notes

In some cases it is not necessary for you to set the active table name because several of the %CPDDUTL control statements (for example, SET TABLE, ADD TABLE, CREATE TABLE, and UPDATE TABLE) set the active table name, based on a specific table reference.

For example, after the %CPDDUTL control statement successfully completes its execution, the following statement sets the active table to the table that is named UCST.

```
ADD TABLE
   NAME=UCST  ... ;
```

Note: The value of the *NAME*= parameter is not case sensitive; thus, NAME=ucst and NAME=UCST are equivalent. △

When a control statement refers to NAME=_ALL_, %CPDDUTL treats that as multiple control statements, one for each table that has a KEPT status of Yes. Because you do not know the order of the tables in the expanded statement and thus you do not know the final value that will be provided for the active table, you can insert a SET TABLE control statement between a statement that refers to _ALL_ tables and a following statement that depends on the value of the active table.

Similarly, if the most recent control statement references a table, it might set the active table to a different table from the one that you want to work on next. Insert a SET TABLE control statement in front of the next control statement to make sure the active table is set to the table that you want.

Also, if there was no earlier reference to a table in the control statement stream, then use a SET TABLE control statement before a control statement, such as DELETE VARIABLE or LIST DERIVED or CREATE FORMULA, that depends on the value of active table.

SET TABLE Example

Use the following statement at any time within your %CPDDUTL job to set the name of the active table to the name *active-table-name*:

```
SET TABLE
   NAME=active-table-name ;
```

STATUS TABLE

Prints the status of a table

STATUS TABLE Overview

Prints in the SAS log the status of a table in the active PDB. Status information includes, for each of the five PDB levels, the following information: the date range, the number of observations, and the age limit.

STATUS TABLE Syntax

STATUS TABLE
NAME=*table-name* | _ALL_;

Details

NAME=*table-name*
is the name of an existing table in the active PDB. Use a specific table name. Or specify _ALL_ to provide the status for all tables in the active PDB.

STATUS TABLE Notes

When this control statement successfully completes its execution, the active table is set to the table that is named in this statement (if the status for a single table is printed) or to the last table (if the status is printed for all tables). If you specify NAME=_ALL_ and you depend on the value of the active table for statements that follow this statement, then use a SET TABLE control statement after this statement to explicitly set the value of the active table.

STOP

Stops the dictionary utility

STOP Overview

This control statement causes the data-dictionary utility to terminate as soon as the control statement is encountered. No following statements are processed and no views are built for any modified tables.

This statement is useful if you are testing your control statements, or if you do not want to rebuild views when this statement is run (because it does not rebuild views for the modified tables). However, if you later decide to rebuild views, run the BUILD VIEWS control statement or rerun your control statements without the STOP.

If you do not code an END, STOP, or QUIT control statement at the end of the %CPDDUTL control statement stream, then an END control statement is implied.

STOP Syntax

STOP;

STOP Notes

The VERIFY SYNTAX control statement also stops views from being built, but unlike the STOP control statement, VERIFY SYNTAX checks the remaining control statements instead of ignoring them.

The STOP control statement is an alias for the QUIT control statement.

SYNCHRONIZE DICTIONARY

Corrects structure errors in the data dictionary

SYNCHRONIZE DICTIONARY Overview

The SYNCHRONIZE DICTIONARY control statement corrects structure errors in the data dictionary of the active PDB.

SYNCHRONIZE DICTIONARY Syntax

SYNCHRONIZE DICTIONARY;

SYNCHRONIZE DICTIONARY Notes

This control statement attempts to rebuild the data dictionary for the current PDB. After you run the SYNCHRONIZE DICTIONARY control statement, run the VERIFY DICTIONARY control statement to ensure that the data dictionary is usable.

Note: Do not attempt to edit the SAS data sets that contain the performance database without using the SAS IT Resource Management GUI or the %CPDDUTL macro. That is, do not use any other SAS tools or external tools to edit the contents of the data dictionary. If you use other SAS tools or external tools, then you might not be able to access the dictionary. If you have modified the PDB's DICTLIB library, then the SYNCHRONIZE DICTIONARY control statement attempts to undo the modifications. If you added extra variables to the data dictionary, then they are deleted. If you removed any variables, then they are restored. However, if you changed the characteristics of any variables, then the undo may fail and you might need to recover a workable version of the data dictionary from a backup copy of the PDB. △

UPDATE DERIVED

Updates a derived variable definition for a table in the active PDB

UPDATE DERIVED Overview

The UPDATE DERIVED control statement updates the definition for a derived variable in the active table in the active PDB. *If a parameter is omitted, then the existing value is retained.* For example, if you specify only the DESCRIPTION= parameter, then only the value of the description changes; the values for other parameters do not change.

Multiple parameters must be separated by at least one space.

UPDATE DERIVED Syntax

UPDATE DERIVED
NAME=*derived-var-name*
<**DAY=**(*statistic-keywords-list*)>
<**WEEK=**(*statistic-keywords-list*)>
<**MONTH=**(*statistic-keywords-list*) >
<**YEAR=**(*statistic-keywords-list*) >
<**DESCRIPTION=**'*string*'>
<**FORMAT=***SAS-format*>
<**INTERPRET=***interpretation*>
<**ISTATS=**(*statistic-keywords-list*)>
<**KEPT=**YES | NO>
<**LABEL=**'*string*'>
<**LENGTH=***number*>
<**SOURCE=**{
source statements
}>
<**SUBJECT=**'*subject-keyword-list*'>
<**VALIDITY=**>
<**WEIGHTVAR=***variable-name*>;

Details

NAME=*derived-var-name*
is the name of the derived variable to be updated. The name must already exist in the table or the CREATE DERIVED control statement for the variable must be included earlier in the stream of control statements so that the derived variable is created before this UDPATE DERIVED control statement is encountered.

DAY=(*statistic-keywords-list*)
WEEK=(*statistic-keywords-list*)
MONTH=(*statistic-keywords-list*)
YEAR=(*statistic-keywords-list*)
is the list of statistic calculations that are requested for this regular variable or derived variable at the specified summary levels for the active table. Use blanks to separate the keywords in the list.

You can use the UPDATE TABLE control statement to specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request (in the CREATE VARIABLE control statement) at each level of the table are calculated for each unique set of values of the class variables.

When creating a new variable definition or derived variable definition, specify all statistics that you want to be calculated at each summary level by using the CREATE VARIABLE or CREATE DERIVED control statement, respectively. If you do not specify statistics for a level, then no statistics are kept at that level. If statistics are specified and if the following are true, then the statistics for that variable are calculated:

- The data type for the variable is numeric.
- The table and variable status are set to KEPT=YES.

- The level where the statistics are to be calculated has an age limit greater than zero.
- The variable is not a CLASS or ID variable at the level where the statistics are to be calculated.

When updating a variable definition or derived variable definition, you do not need to respecify statistics that are already being calculated. For example, if you are already keeping the statistics COUNT and SUM and you also want to keep AVERAGE, then you only need to specify AVERAGE in the UPDATE VARIABLE or the UPDATE DERIVED control statement.

The statistic list can contain one or more of these values at each level:

AVERAGE | NOAVERAGE
for the arithmetic mean

COUNT | NOCOUNT
for the accumulated count of the number of observations that have nonmissing values for the variable, starting at 0

CV | NOCV
for the coefficient of variation

MAXIMUM | NOMAXIMUM
for the maximum value

MINIMUM | NOMINIMUM
for the minimum value

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

DESCRIPTION='*string*'
is a 200-character string that describes the variable. The string must be enclosed in single quotation marks. If the string spans multiple lines, then break it in the middle of a word and continue it in the first column of the next line.

FORMAT=*SAS-format*
is a SAS format to be used with this variable. If the variable's value will contain characters, then assign a character format. If the variable's value will contain numbers, then assign a numeric format. These formats are used for all presentation of the data for this variable.

For more information on default formats that are related to the interpretation type, see the INTERPRET= parameter on the CREATE VARIABLE or CREATE

DERIVED control statement. Also see “Variable Interpretation Types” on page 697. For more information on SAS formats, refer to *SAS Language Reference* for your current release of SAS.

INTERPRET=*interpretation*
indicates the type of information that the variable or derived variable represents. This parameter is required. You can specify one of the following values:

Table 4.4 Values for INTERPRET=

AVERAGE	C2RATE	COUNT
COUNTER	DATE	DATETIME
DECADDRESS	D2RATE	ENUM
FLOAT	FORMULA	GAUGE
HEXFLAGS	INT	IPADDRESS
LABEL	MAXIMUM	MINIMUM
NAME	NETADDRESS	PCTGAUGE
PCTGAUGE100	PERCENT	PERCENT100
RATE	STRING	SUM
TIME	TIMETICKS	UNIXTIME
YNFLAG		

The value can be uppercase or lowercase. For a full description of each value and its defaults, refer to the variable definitions in “Variable Interpretation Types” on page 697. A summary of this information is also available in “Variable Interpretation Type Summary Table” on page 710.

ISTATS=(*statistic-keywords-list*)
lists the default statistics for a variable or derived variable. The statistics are saved with the variable definition, but they are not calculated within the active table. These statistics are used when you run INSTALL TABLE or ADD TABLE control statements, because the master data dictionary does not keep reduction level statistics information.

When called by the UPDATE VARIABLE control statement or the UPDATE DERIVED control statement, ISTATS= applies changes to the default statistics. If an initial statistic is not specified using the UPDATE VARIABLE control statement, the existing setting for that statistic is not changed.

When called by the SET DEFAULTS control statement, ISTATS= sets the default statistics for a variable if you do not set the default statistics when you create the variable.

You can specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request at each level are calculated for each class variable group in each summary level.

The statistics that are calculated for the active table are those that are specified by the DAY=, WEEK=, MONTH=, and YEAR= parameters. Therefore, when you install a table in the master data dictionary, it stores the ISTATS with the table definition. When you later add that table to another PDB, the ADD TABLE statement populates the DAY=, WEEK=, MONTH=, and YEAR= statistics based

on the ISTATS= list of statistics. You can then customize the added table by using the SAS IT Resource Management GUI or an UPDATE VARIABLE/UPDATE DERIVED statement with the DAY=, WEEK=, MONTH=, and YEAR= parameters.

AVERAGE | NOAVERAGE
for the arithmetic mean for all observations

COUNT | NOCOUNT
for an accumulated count of the number of observations that have nonmissing values, starting at zero

CV | NOCV
for the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAXIMUM | NOMAXIMUM
for the maximum value for all observations

MINIMUM | NOMINIMUM
for the minimum value for all observations

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values for all observations

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: ISTATS= can be specified on the SET DEFAULTS, CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, and UPDATE DERIVED statements. If you install or add a table for which ISTATS have not been specified, then the table is given the default statistic values for ISTATS=, which have been specified in SET DEFAULTS for the current set of %CPDDUTL statements. If default values have not been set via SET DEFAULTS, then the variable is treated as if all the statistics in the ISTATS= list were set to NO. If the table is installed and then added, then the added table does not keep any statistics at the day, week, month, or year level unless you set them by using an UPDATE statement. △

If you use a SET DEFAULTS statement for a single variable, then a useful default setting is one that is based on the variable's interpretation. (For more about the relation between interpretation and initial statistics, see "GENERATE SOURCE" on page 604.)

If the SET DEFAULTS statement will apply to a number of variables, then a useful default setting is ISTATS=(AVERAGE).

KEPT=YES | NO
indicates to the process step or reduce step whether data for this variable is to be kept in the PDB. If you specify NO, then the variable's definition is retained in the

dictionary, but existing data is deleted and no new data for this variable is processed or reduced into the PDB. *The default value is NO*.

Note: If you have no current use for the variable, but do anticipate future use, you might want to set the variable's Kept status to *No*. When the variable's Kept status is set to *No*, the variable's definition is retained but ignored, and the variable's values, if any, are discarded. (There will be missing values in all existing observations for the variables that had their Kept status changed from *Yes* to *No*.) △

LABEL='*string*'
is a 40-character string that describes the data in the variable at the detail level. The string must be enclosed in single quotation marks. In the day, week, month, and year levels, the labels are formed automatically by concatenating the statistic name, a period, and the string.

LENGTH=*number*
is the maximum length of the variable as stored in the PDB. *The default value is 0*. Character variables can be as long as 200 characters. The minimum length of a numeric variable is three bytes and its maximum length is eight bytes.

Note: On z/OS, a numeric variable can have a length of two, but that length is not supported in any other operating environments. Therefore, if you specify a length of two, then you cannot report on the data from the SAS IT Resource Management client. △

SOURCE={*source statements*}
specifies the code for the formula variable or derived variable, such as SOURCE={x1=sum(a,b);}. Source statements can be any SAS DATA step statements that result in the creation of a value for the variable that is specified in the NAME= parameter. Remember to end your SAS statements with a semicolon.

Note: Do not omit the braces. If you omit the braces, then you will get various errors. Also, do not make a change in the source code that will cause the variable type to change (such as from numeric to character). △

If you want to create many variables that use the same calculations, then you might want to create a macro that has the common source code for the calculations, add the location of the macro to the SASAUTOS search path, and call the macro in the source statements for each variable.

SUBJECT='*subject-keyword-list*'
specifies a list of keywords that can be used to categorize this variable. The maximum length of the list is 200 characters. Use SUBJECT=' ' in order to leave the list blank. You must use blanks to separate keywords in the list, and the list must be enclosed in single quotation marks.

When updating the SUBJECT= parameter list, you must specify all keywords that you want to use at the time that you update the list, because this parameter does not use pre-existing values.

VALIDITY=
is not currently used.

WEIGHTVAR=*variable-name*
names the alternate variable to be used for statistical weighting. The variable must already exist in the table.

If you do not specify this parameter, then default variable weighting and normalization are used. *By default, analysis variables in INTERVAL tables are weighted and normalized by the DURATION variable during report-time summarization, such as the summarization when Summary Time Period is not AS IS. By default, analysis variables in EVENT tables are not weighted or normalized.*

The following variable interpretation types are normalized (that is, converted to rates for comparison purposes) if the table is an INTERVAL table:

- COUNT
- TIME
- TIMETICKS.

The following variable interpretation types are not normalized (converted) if the table is an INTERVAL table:

- GAUGE
- INT
- MAXIMUM
- MINIMUM.

All other numeric variable interpretation types are weighted if the table is an INTERVAL table.

By default, no weighting or normalization occurs for variables in EVENT tables unless the WEIGHTVAR= parameter is used to specify a weight variable. If a weight variable is specified for a variable in an EVENT table, then the same normalization or weighting rules that are described above are used.

See also "Variable Interpretation Types" on page 697.

UPDATE DERIVED Notes

Some of the regular and/or derived variables in the table can be used as BY, CLASS, ID, or INDEX variables. These variable roles are assigned by UPDATE TABLE, not by CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, or UPDATE DERIVED.

The active table must be set before you can run this statement. To specify the active table, see "SET TABLE" on page 628.

You can use the PRINT TABLE control statement before and after the update, in order to see the existing settings and to verify your changes.

When you use this control statement to update the statistics for a derived variable, the existing statistics are cleared before the update is applied. Therefore, if you specify the DAY=, MONTH=, WEEK=, or YEAR= parameter, then you must specify all statistics that you want to apply to the named variable at that level of the PDB (day, week, month, or year). You need to specify the statistics only for the level that you are updating. For example, if you want to update the statistics at the day level, then you must specify all statistics that you want to apply to that level. This includes any statistics that you want to continue using, as well as any new statistics that you are adding to the specified level. If you are updating statistics for the day level only, then you do not need to specify existing statistics for the other levels.

For more information about the order in which derived and formula variables are calculated, see the section "How to Define Derived and Formula Variables" in the chapter "Administration: Working with Variables" in the SAS IT Resource Management User's Guide.

Note: If KEPT=NO, the UPDATE VARIABLE control statement deletes only the data. The (updated) definition of the derived variable is retained. △

UPDATE FORMULA

Updates the definition of a formula variable in a table

UPDATE FORMULA Overview

The UPDATE FORMULA control statement updates the data dictionary definition of a formula variable in the active table. If a parameter is omitted, then the existing value for that parameter is retained. For example, if you specify only the DESCRIPTION= parameter, then only the value of the description changes; the values for the other parameters remain the same.

Multiple parameters must be separated by at least one space.

UPDATE FORMULA Syntax

UPDATE FORMULA
NAME=*formula-var-name*
<**DESCRIPTION=**'*string*'>
<**FORMAT=***SAS-format*>
<**KEPT=**YES | NO>
<**LABEL=**'*string*'>
<**LEVELS=**'<DETAIL><DAY><WEEK><MONTH><YEAR>'>
<**SOURCE=**{
source statements
}>
<**SUBJECT=**'*subject-keyword-list*'>;

Details

NAME=*formula-var-name*
is the name of the formula variable to be updated. The name must already exist in the table. Or the CREATE FORMULA control statement for this formula variable must be included earlier in the stream of control statements.

DESCRIPTION='*string*'
is a 200-character string that provides information about the formula variable. The string must be enclosed in single quotation marks. If the string spans multiple lines, break it in the middle of a word and continue it in the first column of the next line.

FORMAT=*SAS-format*
is a SAS format to be used with this variable. If the variable's value will contain characters, then assign a character format. If the variable's value will contain numbers, then assign a numeric format. These formats are used for all presentation of the data for this variable.

For more information on default formats that are related to the interpretation type, see the INTERPRET= parameter on the CREATE VARIABLE or CREATE DERIVED control statement. Also see "Variable Interpretation Types" on page 697. For more information on SAS formats, refer to *SAS Language Reference* for your current release of SAS.

KEPT=YES | NO
indicates whether this formula variable is used or ignored. If you specify NO, then the variable's definition is retained in the dictionary, but values are not calculated. If you specify YES (the default), the variable's values are calculated in the view.

LABEL='*string*'

is a 40-character string that describes the formula variable. The string must be enclosed in single quotation marks. This label is used in the view that contains this formula variable.

LEVELS='*levels*'
is a list of levels (DETAIL, DAY, WEEK, MONTH, YEAR) at which the formula variable is to be available. If the formula variable relies on other regular, derived, or formula variables, then those variables and formula variables must exist at all the specified levels. The levels must be enclosed in single quotation marks and separated by one or more spaces, as in LEVELS='DETAIL DAY'.

Note: The formula variable itself, not a statistic that is based on the formula variable, is available at the specified levels. △

SOURCE={*source statements*}
specifies the code for the formula variable or derived variable, such as SOURCE={x1=sum(a,b);}. Source statements can be any SAS DATA step statements that result in the creation of a value for the variable that is specified in the NAME= parameter. Remember to end your SAS statements with a semicolon.

Note: Do not omit the braces. If you omit the braces, then you will get various errors. Also, do not make a change in the source code that will cause the variable type to change (such as from numeric to character). △

If you want to create many variables that use the same calculations, then you might want to create a macro that has the common source code for the calculations, add the location of the macro to the SASAUTOS search path, and call the macro in the source statements for each variable.

SUBJECT='*subject-keyword-list*'
specifies a list of keywords that can be used to categorize this variable. The maximum length of the list is 200 characters. Use SUBJECT=' ' in order to leave the list blank. You must use blanks to separate keywords in the list, and the list must be enclosed in single quotation marks.

When updating the SUBJECT= parameter list, you must specify all keywords that you want to use at the time that you update the list, because this parameter does not use pre-existing values.

Note: You cannot index formula variables. △

UPDATE FORMULA Notes

The active table must be set before you can run this statement. To specify a table as the active table, see "SET TABLE" on page 628.

Formula variables cannot have default statistics at the day, week, month, or year level, because formula variables are created as needed and no data is kept at the levels. If you want statistics at the summary levels, then you must create additional formula variables (at each level for which you want statistics) that are themselves statistics.

For more information on the order in which derived and formula variables are calculated, see the section "How to Define Derived and Formula Variables" in the chapter "Administration: Working with Variables" in the *SAS IT Resource Management User's Guide*.

UPDATE TABLE

Updates a table definition in the active PDB's data dictionary

UPDATE TABLE Overview

The UPDATE TABLE control statement updates the data dictionary definition for an existing table in the active PDB. All the parameters except NAME= are optional. If you specify any optional parameters, then the specified values replace the existing values for those parameters. If parameters are omitted, then the existing values for those parameters are retained.

Note: Multiple parameters must be separated by at least one space. △

UPDATE TABLE Syntax

UPDATE TABLE
NAME=*table-name*
<**ARCHIVE=**YES | NO>
<**COLLECTOR=***collector* | generic>
<**DESCRIPTION=**'*string*'>
<**DETAIL=**(<**AGELIMIT=***number*><**BYVARS=**'*name-list*'><**INDEXVARS=**'*name-list*'>)>
<**DAY=**(<**AGELIMIT=***number*><**CLASSVARS=**'*name-list*'><**IDVARS=**'*name-list*'><**INDEXVARS=**'*name-list*'>)>
<**WEEK=**(<**AGELIMIT=***number*><**CLASSVARS=**'*name-list*'><**IDVARS=**'*name-list*'><**INDEXVARS=**'*name-list*'>)>
<**MONTH=**(<**AGELIMIT=***number*><**CLASSVARS=**'*name-list*'><**IDVARS=**'*name-list*'><**INDEXVARS=**'*name-list*'>)>
<**YEAR=**(<**AGELIMIT=***number*><**CLASSVARS=**'*name-list*'><**IDVARS=**'*name-list*'> <**INDEXVARS=**'*name-list*'>)>
<**EXTNAME=***name*>
<**IDNUM=***number*>
<**KEPT=**YES | NO>
<**LABEL=**'*string*'>
<**TOOLNAME=***name*>;

Details

NAME=*table-name*
is the name of the table definition to be updated. The name must already exist in the active PDB's data dictionary.

ARCHIVE=YES | NO
indicates whether or not your new data for this table should be archived when the data is processed into the PDB. If you want to archive your data when it is processed into the PDB, then specify YES. Archiving of data is "turned on" when you specify ARCHIVE=YES for a table. If ARCHIVE=YES, then when data is processed into this table, the data is also archived.

If you do not want your data archived, then specify NO. *This parameter is not required and the default value is NO*.

COLLECTOR=*collector-name* | *generic*
is the name that (along with the tool name) enables the process task to access and run the correct staging code and use the correct table and variable definitions. The value of the collector name specifies the name of the data collector or data source that creates the log file data. (Which data collectors or data sources are supported by a host depends on the SAS IT Resource Management software license. For

more information, see the SAS IT Resource Management installation instructions.) This is the value that you specify for the COLLECTR= parameter on the process macro (%CxPROCES) when you process your data into the PDB.

Note: %CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CWPROCES. △

For more about the value to use for *collector-name*, see the Notes section for the UPDATE TABLE control statement.

If the name of the collector contains special characters, then enclose the name in single quotation marks. Here is an example, if you are using HP-OVPA:

```
collector='hp-ovpa'
```

DESCRIPTION='*string*'
is a 200-character string that you can use to describe the data in the table. The string must be enclosed in single quotation marks. If the string spans multiple lines, break it in the middle of a word and continue in the first column of the next line.

DETAIL=(AGELIMIT=*number*)
DAY=(AGELIMIT=*number*)
WEEK=(AGELIMIT=*number*)
MONTH=(AGELIMIT=*number*)
YEAR=(AGELIMIT=*number*)
defines the age range that the data can have at each PDB level. Existing data in each level of the PDB is "aged out" (removed) when the range of DATETIME reaches the value specified in the AGELIMIT parameter. The default value and representation of the number value at each level are

- DETAIL - *number* represents days; the default is 10 days.
- DAY - *number* represents days; the default is 45 days.
- WEEK - *number* represents weeks; the default is 15 weeks.
- MONTH - *number* represents months; the default is 18 months.
- YEAR - *number* represents years; the default is 5 years.

The *number* must be an integer.

If AGELIMIT=0 at the detail level, your data is stored in the detail level until the next reduce or process task runs. Therefore, if the reduce task runs immediately after the process task, the data is processed, reduced into the summary levels, and then removed from the detail level. However, if the process task runs again before the data has been reduced, the existing data in the detail level is overwritten and, thus, not incorporated in the statistics at the summary levels.

Using AGELIMIT=0 at the detail level is useful if you are collecting and processing large amounts of data on a daily basis; however, to avoid losing data at summary levels, you should always reduce the data immediately after you process it. Using AGELIMIT=0 at any other level means that no data is kept at that level.

At the week, month, or year level, if AGELIMIT=*number* then data is kept for the most recent partial week, month, or year and for the previous *number-1* full weeks, months, or years. In other words, AGELIMIT=*number* means up to but not including *number* full weeks, months, or years of data.

Note: For information on setting age limits for your data, see the section "Specifying the Age Limit for a Level in a Table" in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*. △

DETAIL=(BYVARS='*name-list*')
is a list of regular and/or derived variables by which the process task merges data into the PDB. The order of the variables determines the sort order of the table's

existing data at the detail level and of the table's incoming data. (If the data is already in that order, no sorts are executed.) Items in the list must be separated by blanks. The list must be enclosed in single quotation marks and cannot exceed 200 characters in length. The order of the variables determines the sort order of the table's data at the detail level and is the default sort order for reports that are based on the data.

Only variables that are already defined in this table at the time this statement is encountered can be used in the list. The DATETIME variable must be in the list. The data is sorted in the new order when the views are rebuilt, which occurs at the end of the %CPDDUTL control statement stream (unless the stream includes a STOP or QUIT control statement, in which case the views are not rebuilt) or when the next process or reduce task runs, whichever comes first. Formula variables are not allowed in the BY variables list.

If observations with duplicate values of BY variables are detected, then information about the number of observations with duplicate BY variables is printed in the summary report that is produced each time the process task runs.

For more information on modifying the list of BY variables, see the section "Restrictions on Modifications of BY Variables Lists" section in the chapter "Administration: Working with Levels" in the *SAS IT Resource Management User's Guide*.

DETAIL=(INDEXVARS='*name-list*')
DAY=(INDEXVARS='*name-list*')
WEEK=(INDEXVARS='*name-list*')
MONTH=(INDEXVARS='*name-list*')
YEAR=(INDEXVARS='*name-list*')

specifies a list of regular and/or derived variables for each level of the PDB. The variables are used to define a set of simple indexes, one index for each variable that is named in the list. Only variables that are already in the table can be used in the list. Selecting index variables can lead to improved performance in reporting.

For the day, week, month, and year levels, each variable that is named in the list must be either a CLASS or ID variable.

The list must be enclosed in quotation marks and cannot exceed 200 characters in length.

A *simple index variable* is a variable for which an index is built; for example, the variable MACHINE is often used as an index variable. The index contains information on the values of the variable and the observations on which the values reside.

Restrictions on the use of index variables are as follows:

1. You cannot index formula variables.
2. Any (non-formula) variable at the detail level can be indexed.
3. Only CLASS or ID variables at the day, week, month, and year levels can be indexed.

DAY=(CLASSVARS='*name-list*')
WEEK=(CLASSVARS='*name-list*')
MONTH=(CLASSVARS='*name-list*')
YEAR=(CLASSVARS='*name-list*')

is a list of regular and/or derived variables that are used to classify or group the data for each summary level. Each level can contain different variables. The order of the variables determines the sort order of the table's data at the specified level and is the default sort order for reports that are based on the data in the level.

The lists must be enclosed in single quotation marks and cannot exceed 200 characters in length. Use blanks to separate the variables in the list, and name the variables as they are named in the detail level. Only variables that are already defined in this table when this statement is encountered can be used in the list.

Formula variables are not allowed in the list. The DATETIME variable must be in the list. The data is sorted in the new order when the views are rebuilt, which occurs at the end of the %CPDDUTL control statements stream (unless the stream includes a STOP or QUIT control statement, in which case the views are not rebuilt) or when the next process or reduce task runs, whichever comes first.

CAUTION:

If you are using the PATROL data collector, then do not modify the CLASS list or you might encounter problems when you process the data. Specifically, you should not remove the DURGRP variable from your CLASS variable list. The DURGRP variable ensures that your data is summarized correctly at each of the summary levels. △

In this statement, do not list statistics that are based on the variables. For more information, see the statistics keyword list in the following %CPDDUTL control statements: "CREATE VARIABLE" on page 591, "UPDATE VARIABLE" on page 645, "CREATE DERIVED" on page 577, and "UPDATE DERIVED" on page 631.

DAY=(IDVARS='*name-list*')
WEEK=(IDVARS='*name-list*')
MONTH=(IDVARS='*name-list*')
YEAR=(IDVARS='*name-list*')

is a list of the regular and derived variables that are to be used as ID variables for each summary level. Typically, an ID variable is a "twin" of one of the CLASS variables, and its values are used in reports as another way of identifying the same data. The lists must be enclosed in single quotation marks and cannot exceed 200 characters in length. Use blanks to separate the variables in the list. Only variables that are already defined in this table when this statement is encountered can be used in the list.

ID variables are used for identification rather than to summarize data. The value of an ID variable is the last value that is presented to the particular class (as determined by the values of the CLASS variables) for the given summary level. In the situation where data is processed and reduced in nonascending DATETIME order (that is, backloading data), the value of the ID variable is the most recent value in that class to arrive. However, because it is old data that is arriving, the value of the ID variable might not be the most recent value in that class when more current data is also taken into account.

An example of an ID variable would be the volume label for a disk. While the physical device identifier for the disk can be used as a CLASS variable, the volume label for the disk is more useful for reporting purposes.

Because the value of an ID variable is the "last presented" value, use of ID variables at the levels that summarize a longer time period (such as month or year) might not prove useful. Using the example above, if the disk label changes on the twelfth of December, the year level of the PDB will record the new value for the ID variable, which does not reflect the label for the majority of the year.

EXTNAME=*name*

is the name of the external data source from which data is processed into the detail level of the PDB. The data source could be a SAS data set or an external file, depending on the data collector or data source from which you obtain the data.

For more about the value to use for external-name, see the Notes section for the UPDATE TABLE control statement.

The name can be a maximum of eight characters long on z/OS or 200 characters long on other operating environments. The name must begin with a letter, can contain letters and numbers, and can contain blanks or special characters. The name is not case sensitive.

If you do not specify this parameter, then a blank value is stored in the data dictionary. The variable name field is used to determine what variable is to be processed. That is, when the next process task runs, the *blank* external name defaults to the value that is specified in the NAME= parameter.

IDNUM=*number*
specifies a numeric identifier that has different uses for each collector. Refer to the information about a specific collector in the *SAS IT Resource Management Server Setup Guide*.

This parameter is used only for UNIX and Windows.

KEPT=YES | NO
indicates to the process task whether this table is to be used or ignored. If you specify YES, then the table definition and existing data at any level of the PDB are retained. New data for this table is processed into the detail level of the PDB, and any existing data in the detail level is processed into the summary levels of the PDB. If you specify NO, then the table definition and existing data at any level of the PDB are retained in the data dictionary, but new data for this table is not processed into the detail level of the PDB and any existing data in the detail level is not processed to the summary levels of the PDB. *The default value is NO.*

The KEPT= parameter is ignored when you specify a list of tables in any of the %C*x*PROCESS macros, where *x* is M, P, S, or W.

LABEL='*string*'
is a 40-character string describing the data in the table. The string must be enclosed in single quotation marks. This label is used in the views created for this table.

TOOLNAME=*name*
is the name that (along with the collector-name) enables the process task to access and run the correct staging code and to use the correct table and variable definitions.

For more about the value to use for tool-name, see the Notes section for the UPDATE TABLE control statement.

This name cannot exceed 32 characters.

UPDATE TABLE Notes

When this control statement successfully completes its execution, the active table is set to the table named in this control statement.

You cannot directly change the table type (Interval/Event) after the table is created. If you want to change the table type, then use the GENERATE SOURCE control statement to output the existing table definition. Then, revise the type in the output, delete the existing table definition, and use the edited output as an input control statement stream for %CPDDUTL in order to create the table with the revised type.

Regular and/or derived variables in the table can be used as BY, CLASS, ID, or INDEX variables. These variable roles are assigned by UPDATE TABLE.

The COLLECTOR=, TOOLNAME=, and EXTNAME= parameters operate as follows:

- For tables that SAS IT Resource Management supplies, do not change the collector name, tool name, or external name.

  For more information, see the documentation for any of the process macros, such as "%CPPROCES" on page 154.

  The only place that the external name is documented is in the table definition.

  When you process data into this table with a process macro, specify

```
%CxPROCES(...COLLECTR=collector-name,TOOLNM=tool-name,...);
```

For more information about the %CMPROCES macro, see "%CMPROCES" on page 41. For more information about the %CPPROCES macro, see "%CPPROCES" on page 154. For more information about the %CSPROCES macro, see "%CSPROCES" on page 204. For more information about the %CWPROCES macro, see "%CWPROCES" on page 220.

- For user-created tables, if you *have* created the table definition as part of a set of collector-support entities for installation and use with %CPPROCES,
  - the collector name and tool name should be the same as the ones that you have used or will use to "tag" the package. For more information about collector-support entity packages and installation, see the section "Collector Support Packages" in the chapter "Administration: Extensions to SAS IT Resource Management" in the *SAS IT Resource Management User's Guide*.

    When you process data into this table with a process macro, specify

    ```
    %CPPROCES(...COLLECTR=collector-name,TOOLNM=tool-name,...);
    ```

  - the external name should be the name of the SAS data set or view (in the COLLECT library) that represents the staged data.

    For example, if the staging code stages the data to COLLECT.XYZ, then XYZ is the external name.
- For user-created tables, if you *have not* created the table definition as part of a set of collector-support entities for installation and use with %CPPROCES,
  - use GENERIC as the collector name, and use SASDS or CHARDELIM as the tool name. (If you want to use the supplied staging code for character-delimited data, specify CHARDELIM. Otherwise, specify SASDS.)

    When you process data into this table with a process macro, specify

    ```
    %CxPROCES(...COLLECTR=GENERIC,TOOLNM=tool-name,...)
    ```

  - the external name should be the name of the SAS data set or view (in the COLLECT library) that represents the staged data.

    For example, if the staging code stages the data to COLLECT.XYZ, then XYZ is the external name.

UPDATE VARIABLE

Updates the definition of a regular variable in a table

UPDATE VARIABLE Overview

The UPDATE VARIABLE control statement updates the data dictionary definition for a regular variable in the active table in the active PDB. *If a parameter is omitted, the existing value is retained.* For example, if you specify only the DESCRIPTION= parameter, only the value of the description changes; the values for other parameters do not change.

Multiple parameters must be separated by at least one space.

UPDATE VARIABLE Syntax

UPDATE VARIABLE
NAME=*variable-name*

<**DAY=**(*statistic-keywords-list*)>
<**WEEK=**(*statistic-keywords-list*)>
<**MONTH=**(*statistic-keywords-list*)>
<**YEAR=**(*statistic-keywords-list*)>
<**DESCRIPTION=**'*string*'>
<**EXTNAME=***name*>
<**FORMAT=***SAS-format*>
<**IDNUM=***number*>
<**INFORMAT=**>
<**INTERPRET=***interpretation*>
<**ISTATS=**(*statistic-keywords-list*)>
<**KEPT=**YES | NO>
<**LABEL=**'*string*'>
<**LENGTH=***number*>
<**OID=***oid-specification*>
<**SUBJECT=**'*subject-keyword-list*'>
<**VALIDITY=**>
<**WEIGHTVAR=***variable-name*>;

Details

NAME=*variable-name*
is the name of the regular variable to be updated. The name must already exist in the table, or the CREATE VARIABLE control statement for the variable must be included earlier in the stream of control statements.

DAY=(*statistic-keywords-list*)
WEEK=(*statistic-keywords-list*)
MONTH=(*statistic-keywords-list*)
YEAR=(*statistic-keywords-list*)
is the list of statistic calculations that are requested for this regular variable or derived variable at the specified summary levels for the active table. Use blanks to separate the keywords in the list.

You can use the UPDATE TABLE control statement to specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request (in the CREATE VARIABLE control statement) at each level of the table are calculated for each unique set of values of the class variables.

When creating a new variable definition or derived variable definition, specify all statistics that you want to be calculated at each summary level by using the CREATE VARIABLE or CREATE DERIVED control statement, respectively. If you do not specify statistics for a level, then no statistics are kept at that level. If statistics are specified and if the following are true, then the statistics for that variable are calculated:

- The data type for the variable is numeric.
- The table and variable status are set to KEPT=YES.
- The level where the statistics are to be calculated has an age limit greater than zero.
- The variable is not a CLASS or ID variable at the level where the statistics are to be calculated.

When updating a variable definition or derived variable definition, you do not need to respecify statistics that are already being calculated. For example, if you

are already keeping the statistics COUNT and SUM and you also want to keep AVERAGE, then you only need to specify AVERAGE in the UPDATE VARIABLE or the UPDATE DERIVED control statement.

The statistic list can contain one or more of these values at each level:

AVERAGE | NOAVERAGE
for the arithmetic mean

COUNT | NOCOUNT
for the accumulated count of the number of observations that have nonmissing values for the variable, starting at 0

CV | NOCV
for the coefficient of variation

MAXIMUM | NOMAXIMUM
for the maximum value

MINIMUM | NOMINIMUM
for the minimum value

NMISS | NONMISS
for the number of observations with missing variable values

RANGE | NORANGE
for the difference between MAXIMUM and MINIMUM

STD | NOSTD
for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
for the sum of values

USS | NOUSS
for the uncorrected sum of squares

VARIANCE | NOVARIANCE
for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

DESCRIPTION='*string*'
is a 200-character string that describes the variable. The string must be enclosed in single quotation marks. If the string spans multiple lines, then break it in the middle of a word and continue it in the first column of the next line.

EXTNAME=*name*
is the name of the variable as it appears in the external data source from which data is processed into the detail level of the PDB. The data source could be a SAS data set or an external file, depending on the collector that you used to collect the data.

If COLLECTOR=GENERIC for this table, then EXTNAME= must be the name of the variable as it exists in the SAS data set or view. The variable name must be a valid SAS name. It can be a maximum of eight characters long, must begin with a letter, can contain letters and numbers, and cannot contain blanks or special characters. The name is not case sensitive.

If you do not specify this parameter, then a blank value is stored in the data dictionary. The variable name field is used to determine what variable is to be processed. That is, when the next process task runs, the *blank* external name defaults to the value that is specified in the NAME= parameter.

FORMAT=*SAS-format*
is a SAS format to be used with this variable. If the variable's value will contain characters, then assign a character format. If the variable's value will contain numbers, then assign a numeric format. These formats are used for all presentation of the data for this variable.

For more information on default formats that are related to the interpretation type, see the INTERPRET= parameter on the CREATE VARIABLE or CREATE DERIVED control statement. Also see "Variable Interpretation Types" on page 697. For more information on SAS formats, refer to *SAS Language Reference* for your current release of SAS.

IDNUM=*number*
specifies a numeric identifier that has different uses for each collector. Refer to the information about a specific collector in the *SAS IT Resource Management Server Setup Guide*.

This parameter is used only for UNIX and Windows.

INFORMAT=
is not currently used.

INTERPRET=*interpretation*
indicates the type of information that the variable or derived variable represents. This parameter is required. You can specify one of the following values:

Table 4.5 Values for INTERPRET=

AVERAGE	C2RATE	COUNT
COUNTER	DATE	DATETIME
DECADDRESS	D2RATE	ENUM
FLOAT	FORMULA	GAUGE
HEXFLAGS	INT	IPADDRESS
LABEL	MAXIMUM	MINIMUM
NAME	NETADDRESS	PCTGAUGE
PCTGAUGE100	PERCENT	PERCENT100
RATE	STRING	SUM
TIME	TIMETICKS	UNIXTIME
YNFLAG		

The value can be uppercase or lowercase. For a full description of each value and its defaults, refer to the variable definitions in "Variable Interpretation Types" on page 697. A summary of this information is also available in "Variable Interpretation Type Summary Table" on page 710.

ISTATS=(*statistic-keywords-list*)
lists the default statistics for a variable or derived variable. The statistics are saved with the variable definition, but they are not calculated within the active table. These statistics are used when you run INSTALL TABLE or ADD TABLE control statements, because the master data dictionary does not keep reduction level statistics information.

When called by the UPDATE VARIABLE control statement or the UPDATE DERIVED control statement, ISTATS= applies changes to the default statistics. If

an initial statistic is not specified using the UPDATE VARIABLE control statement, the existing setting for that statistic is not changed.

When called by the SET DEFAULTS control statement, ISTATS= sets the default statistics for a variable if you do not set the default statistics when you create the variable.

You can specify a class variable list for each level of a table in the PDB. For example, you might specify MACHINE and SHIFT as class variables for a specific level in a table. The data at each level in the table is grouped according to the values of the class variables. The statistics that you request at each level are calculated for each class variable group in each summary level.

The statistics that are calculated for the active table are those that are specified by the DAY=, WEEK=, MONTH=, and YEAR= parameters. Therefore, when you install a table in the master data dictionary, it stores the ISTATS with the table definition. When you later add that table to another PDB, the ADD TABLE statement populates the DAY=, WEEK=, MONTH=, and YEAR= statistics based on the ISTATS= list of statistics. You can then customize the added table by using the SAS IT Resource Management GUI or an UPDATE VARIABLE/UPDATE DERIVED statement with the DAY=, WEEK=, MONTH=, and YEAR= parameters.

AVERAGE | NOAVERAGE
: for the arithmetic mean for all observations

COUNT | NOCOUNT
: for an accumulated count of the number of observations that have nonmissing values, starting at zero

CV | NOCV
: for the coefficient of variation. It is calculated as the standard deviation divided by the mean and multiplied by 100.

MAXIMUM | NOMAXIMUM
: for the maximum value for all observations

MINIMUM | NOMINIMUM
: for the minimum value for all observations

NMISS | NONMISS
: for the number of observations with missing variable values

RANGE | NORANGE
: for the difference between MAXIMUM and MINIMUM

STD | NOSTD
: for the standard deviation (square root of variance). Like the variance, it is a measure of the dispersion about the mean but in the same unit of measure as the data.

SUM | NOSUM
: for the sum of values for all observations

USS | NOUSS
: for the uncorrected sum of squares

VARIANCE | NOVARIANCE
: for a measure of the dispersion or variability of the data about the mean. When values are close to the mean, the variance is small. When values are scattered widely about the mean, the variance is large.

Note: ISTATS= can be specified on the SET DEFAULTS, CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, and UPDATE DERIVED statements. If you install or add a table for which ISTATS have not been specified, then the

table is given the default statistic values for ISTATS=, which have been specified in SET DEFAULTS for the current set of %CPDDUTL statements. If default values have not been set via SET DEFAULTS, then the variable is treated as if all the statistics in the ISTATS= list were set to NO. If the table is installed and then added, then the added table does not keep any statistics at the day, week, month, or year level unless you set them by using an UPDATE statement. △

If you use a SET DEFAULTS statement for a single variable, then a useful default setting is one that is based on the variable's interpretation. (For more about the relation between interpretation and initial statistics, see "GENERATE SOURCE" on page 604.)

If the SET DEFAULTS statement will apply to a number of variables, then a useful default setting is ISTATS=(AVERAGE).

KEPT=YES | NO
indicates to the process step or reduce step whether data for this variable is to be kept in the PDB. If you specify NO, then the variable's definition is retained in the dictionary, but existing data is deleted and no new data for this variable is processed or reduced into the PDB. *The default value is NO*.

Note: If you have no current use for the variable, but do anticipate future use, you might want to set the variable's Kept status to *No*. When the variable's Kept status is set to *No*, the variable's definition is retained but ignored, and the variable's values, if any, are discarded. (There will be missing values in all existing observations for the variables that had their Kept status changed from *Yes* to *No*.) △

LABEL='*string*'
is a 40-character string that describes the data in the variable at the detail level. The string must be enclosed in single quotation marks. In the day, week, month, and year levels, the labels are formed automatically by concatenating the statistic name, a period, and the string.

LENGTH=*number*
is the maximum length of the variable as stored in the PDB. *The default value is 0*. Character variables can be as long as 200 characters. The minimum length of a numeric variable is three bytes and its maximum length is eight bytes.

Note: On z/OS, a numeric variable can have a length of two, but that length is not supported in any other operating environments. Therefore, if you specify a length of two, then you cannot report on the data from the SAS IT Resource Management client. △

OID=*oid-specification*
is an SNMP OID (object ID) of the form *n.n.n.n.n.n*. The value must be unique for the variable, if you specify a value. Use *OID=.* to specify none.

This parameter is used only on UNIX.

SUBJECT='*subject-keyword-list*'
specifies a list of keywords that can be used to categorize this variable. The maximum length of the list is 200 characters. Use SUBJECT=' ' in order to leave the list blank. You must use blanks to separate keywords in the list, and the list must be enclosed in single quotation marks.

When updating the SUBJECT= parameter list, you must specify all keywords that you want to use at the time that you update the list, because this parameter does not use pre-existing values.

VALIDITY=
is not currently used.

WEIGHTVAR=*variable-name*

names the alternate variable to be used for statistical weighting. The variable must already exist in the table.

If you do not specify this parameter, then default variable weighting and normalization are used. *By default, analysis variables in INTERVAL tables are weighted and normalized by the DURATION variable during report-time summarization, such as the summarization when Summary Time Period is not AS IS. By default, analysis variables in EVENT tables are not weighted or normalized.*

The following variable interpretation types are normalized (that is, converted to rates for comparison purposes) if the table is an INTERVAL table:

- COUNT
- TIME
- TIMETICKS.

The following variable interpretation types are not normalized (converted) if the table is an INTERVAL table:

- GAUGE
- INT
- MAXIMUM
- MINIMUM.

All other numeric variable interpretation types are weighted if the table is an INTERVAL table.

By default, no weighting or normalization occurs for variables in EVENT tables unless the WEIGHTVAR= parameter is used to specify a weight variable. If a weight variable is specified for a variable in an EVENT table, then the same normalization or weighting rules that are described above are used.

See also “Variable Interpretation Types” on page 697.

UPDATE VARIABLE Notes

Some of the regular and/or derived variables in the table can be used as BY, CLASS, ID, or INDEX variables. These variable roles are assigned by UPDATE TABLE, not by CREATE VARIABLE, UPDATE VARIABLE, CREATE DERIVED, or UPDATE DERIVED.

The active table must be set before you can run this statement. To specify a table as the active table, see “SET TABLE” on page 628.

If you want to stop using the variable but do not want to delete the definition and the data, then use the UPDATE VARIABLE control statement with KEPT=NO. Data will no longer be processed or reduced for the variable, but the definition and the existing data will still persist until it is aged out.

You can use the PRINT TABLE control statement before and after this statement, in order to see the active settings and verify your changes.

When you use this control statement to update the statistics for a variable, the active statistics are cleared before the update is applied. Therefore, if you specify the DAY=, MONTH=, WEEK=, or YEAR= parameter, then you must specify all statistics that you want to apply to the named variable at that level of the PDB (day, week, month, or year). You need to specify the statistics only for the level that you are updating. For example, if you want to update the statistics at the day level, then you must specify all statistics that you want to apply to that level. This includes any active statistics that you want to continue using, as well as any new statistics that you are adding to the specified level. If you are updating statistics only for the day level, then you do not need to specify existing statistics for the other levels.

UPDATE VARIABLE Example

This example modifies a variable in a table in the active PDB.

```
/*  NB: - Needs update access to the active PDB           */
/*      - This statement stream                           */
/*           sets the active table name                  */
/*           modifies a variable in the table.           */
/*      - The UPDATE VARIABLE control statement changes the       */
/*        setting of three initial statistics.  The       */
/*        other initial statistics retain the settings    */
/*        they had.                                       */

set table
  name=TABXYZ;
update variable
  name=uuxx
  description='Site-specific special variable - ver 2'
  istats=(MAXIMUM NORANGE AVERAGE)
  ;
```

VERIFY DICTIONARY

Checks the format of the data dictionary

VERIFY DICTIONARY Overview

The VERIFY DICTIONARY control statement checks the structure and integrity of the data dictionary in the active PDB and reports any errors. Error messages are printed in the output window or the SAS log.

VERIFY DICTIONARY Syntax

VERIFY DICTIONARY;

VERIFY DICTIONARY Notes

The VERIFY DICTIONARY control statement examines the data dictionary more extensively than the %CPSTART macro does. The usability of the dictionary is automatically verified (by %CPSTART) each time you start the %CPSTART macro. It is also verified (by %CPSTART) each time you start the SAS IT Resource Management GUI or switch the active PDB within the GUI.

CAUTION:

Edit the SAS data sets that contain the performance database by using only the SAS IT Resource Management GUI or the SAS IT Resource Management macros. Do not use any other SAS tools or external tools to edit the contents of these data sets. If you use other SAS tools or external tools, then you might not be able to access the dictionary later. For more information, refer to the SYNCHRONIZE DICTIONARY control statement. △

VERIFY SYNTAX

Runs a job in order to check the %CPDDUTL syntax

VERIFY SYNTAX Overview

The VERIFY SYNTAX statement turns on the syntax-check mode and turns off the execution mode. This statement applies to your current control statement stream of %CPDDUTL.

VERIFY SYNTAX Syntax

VERIFY SYNTAX;

VERIFY SYNTAX Notes

This control statement checks whether the syntax in the %CPDDUTL control statements that follow VERIFY SYNTAX is correct. The VERIFY SYNTAX control statement does not update the data dictionary in the active PDB. VERIFY SYNTAX control statement does not discover any errors relating to the data dictionary because the data dictionary is not accessed during syntax-check mode.

XREF TABLE

Prints a formula listing for a table

XREF TABLE Overview

The XREF TABLE control statement prints a list of formula variables in the specified table in the active PDB. For each formula variable, it prints a list of the variables that are used to construct the formula. The list is printed in the SAS log.

XREF TABLE Syntax

XREF TABLE
    NAME=*table-name* | _ALL_ ;

Details

NAME=*table-name* | _ALL_
  is the name of an existing table in the active PDB. Use a specific table name or specify _ALL_ in order to provide a listing for all tables in the active PDB.

XREF TABLE Notes

When this control statement successfully completes its execution, the active table is set to the table that is named in this statement (if a cross reference is run for a single

table) or to the last table that is referenced (if a cross reference is run for all tables). If you specify NAME=_ALL_ and you depend on the value of the active table for statements that follow this statement, then use a SET TABLE statement after this statement in order to explicitly set the value of the active table.

To print the source statements that are used to calculate the values of the formula variables and derived variables, use the PRINT TABLE control statement.

APPENDIX

1

Recovery Procedures

Troubleshooting Batch Jobs That Fail

The following guidelines will help you to locate and solve problems that occur when you run batch jobs. If a batch job fails, then it will be important to have access to backups of your PDB. It is a good idea to back up your PDB every day and to retain several days' worth of backups. Also, you should always check the results of your batch job so that you are aware of any problems.

If a batch job fails, then the first thing that you should do is to check the SAS log for information about the reason for failure. You should then fix that problem before rerunning the job. For standard information on recovering from a failure, refer to the recovery information below for the specific macro that you are using.

If you cannot solve the problem by using this information, then contact SAS Technical Support. Be ready to provide your general site information (site number, company name, your name and phone number, product that you are using, product release, operating system on which you are running, and type of host). Also be ready to provide a copy of the complete SAS log that indicates what problems occurred.

The following guidelines are available for troubleshooting batch jobs that fail:

- "%CPSTART Recovery Procedures" on page 656
- "%CxPROCES Macros Recovery Procedures" on page 656
- "%CPREDUCE Recovery Procedures" on page 657
- "Reporting Macros Recovery Procedures" on page 657
- "Recovery Procedures for Other Macros" on page 658.

Also, the following guidelines may be useful during recovery:

- "Guidelines for Allocating Space in the PDB" on page 662
- "Guidelines for Revising Space Allocation in the PDB" on page 664
- "Guidelines for Allocating Space in the Archive" on page 665

□ “Guidelines for Revising Space Allocations in the Archive” on page 665.

%CPSTART Recovery Procedures

If the macro is creating a PDB and fails for any reason, then delete the partially created PDB, check for error messages in the SAS log, fix the problem, and resubmit at least the %CPSTART macro.

If the macro is not creating a PDB and the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the %CPSTART macro.

If the macro is not creating a PDB and the macro completes its execution, but the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log, fix the problem, and resubmit at least the %CPSTART macro.

%CxPROCES Macros Recovery Procedures

In the following instructions, %CxPROCES refers to the process macros %CMPROCES (for z/OS), %CPPROCES (portable), %CSPROCES (for UNIX), and %CWPROCES (for Windows).

If the macro fails before completion because of an out-of-space condition while staging data, then fix the problem and resubmit at least the %CxPROCES macro and the preceding %CPSTART macro.

If the macro fails before completion because you ran out of space while processing staged data into the detail level of the PDB, then do the following:

1. If you are using %CMPROCES or %CPPROCES without duplicate checking enabled, then restore the PDB from the backup that you made after the previous run of process and reduce. (If you are using %CMPROCES or %CPPROCES with duplicate checking enabled, or if you are using %CSPROCES or %CWPROCES, then duplicate checking prevents duplicate observations from being processed into the PDB. In this case you do not need to restore the PDB to its previous state.)
2. For all the %CxPROCES macros, change the allocations for the PDB (see the note below that is related to estimating space in the PDB), delete the archive library that the failed run created (if the run got that far and if archiving was requested), and resubmit at least the %CxPROCES macro and the preceding %CPSTART macro.

If the macro fails before completion because of an out-of-space condition in the run’s archive library, then do the following:

1. If you are using %CMPROCES or %CPPROCES without duplicate checking enabled, then restore the PDB from the backup that you made after the previous run of process and reduce. (If you are using %CMPROCES or %CPPROCES with duplicate checking enabled or if you are using %CSPROCES or %CWPROCES, then duplicate checking prevents duplicate observations from being processed into the PDB. In this case you do not need to restore the PDB to its previous state.)
2. For all the %CxPROCES macros, delete the archive library that the failed run created, change the allocations for the archive (see “Guidelines for Allocating Space in the Archive” on page 665 and “Guidelines for Revising Space Allocations in the Archive” on page 665), and resubmit at least the %CxPROCES macro and the preceding %CPSTART macro.

If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log,

restore the PDB from the backup that you made after the previous run of process and reduce, fix the problem, delete the archive library that the failed run created, and resubmit at least the %CxPROCES macro and the preceding %CPSTART macro. (Note that in some cases the PDB might not need to be restored and the archive library might not need to be deleted. When in doubt, however, restore and delete.)

It is a good idea to do preventive maintenance on the space allocations for the PDB and archive. It is more convenient to address space issues before a problem happens than to wait until the process fails. (See "Guidelines for Allocating Space in the PDB" later in this appendix and "Guidelines for Revising PDB Library Allocations" later in this appendix.)

%CPREDUCE Recovery Procedures

If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the %CPREDUCE macro and the preceding %CPSTART macro. The %CPREDUCE macro resumes reduction from the most recent checkpoint.

If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log, fix the problem, and resubmit at least the %CPREDUCE macro and the preceding %CPSTART macro. The %CPREDUCE macro resumes reduction from the most recent checkpoint.

In both of the cases above, if the macro fails in that table again, then use one of the following methods to solve the problem so that you can continue. If you choose method 2, then you can still use method 1 later.

- Method 1: Restore the PDB from the backup that you made after the most recent process-and-reduce run, find and fix the problem, and submit a job to start SAS IT Resource Management and to process and reduce the data that was collected since the time of the backup.
- Method 2: In the table definition of the table in which the error occurred, set the Kept status to *NO*. (See the UPDATE TABLE control statement for the %CPDDUTL macro.) Then resubmit at least the %CPSTART and %CPREDUCE macros, this time with the CKPTINIT= parameter set to *YES* on the %CPREDUCE macro. Because the CKPTINIT= parameter is set to *YES*, the information that was being held in the checkpoint data set for that table is lost and thus is not available to reduce into the summary levels of that table. Because the table's Kept status is *NO*, the table is not available for reduction or processing. (After you investigate and fix the problem, you can set the table's Kept status to *YES* and then use method 1 for a complete recovery.)

Reporting Macros Recovery Procedures

This section refers to the following reporting macros: %CPCCHRT, %CPCHART, %CPG3D, %CPPLOT1, %CPPLOT2, %CPPRINT, %CPRUNRPT, %CPSPEC, %CPTABRPT, and %CPXHTML. For recovery information related to other SAS IT Resource Management report and administration macros, see "Recovery Procedures for Other Macros" on page 658 .

If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem) or if the macro completes its execution but writes an error message to the SAS log, then do one of the following:

- If the macro does not specify OUTMODE=WEB, then fix the problem and resubmit at least the reporting macro and the preceding %CPSTART macro.

- If the macro specifies OUTMODE=WEB, then use the %CPWEBINI macro (for the %CPRUNRPT and %CPXHTML macros, you can use the WEBCLR=YES parameter as an alternative to %CPWEBINI) or the %CPMANRPT macro (for the %CPRUNRPT macro) to delete all or some specified reports in the Web catalog and Web directory/ies to which the failed macro was writing. Then fix the problem and resubmit at least the reporting macro and the preceding %CPSTART macro. Also, resubmit the macros that generated the other reports that were deleted. (When regenerating the other reports, you might need to modify the datetime subsetting criteria so that the report that is generated at this time matches the report that was generated earlier.)

Recovery Procedures for Other Macros

- *For %CMFTPSND*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the %CMFTPSND macro and the preceding %CPSTART macro.

  If the macro writes a warning message because of lack of input, then the macro variable that is specified by the _RC= parameter has a return code of -1. See the CNTLPFX= parameter for a note about the FTPCNTL and TCPSEND members in the PDSs. Fix the problem and resubmit at least the %CMFTPSND macro and the preceding %CPSTART macro.

  If the macro writes an error message because of an FTP problem, then the macro variable that is specified by the _RC= parameter has a return code that is a copy of the FTP program's return code. See your FTP's documentation for more information about the return code. Fix the problem and resubmit at least the %CMFTPSND macro and the preceding %CPSTART macro.

  If the macro returns neither a warning nor an error message, then the FTP program did not report any problem. The macro variable that is specified by the _RC= parameter has a return code of 0. *In the case of this macro, a return code of 0 does not guarantee that FTP executed all commands successfully. Different FTP programs behave differently. It is possible that your FTP program encountered a problem and did not report it to the macro. The first time that you submit this macro or submit a change in this macro or submit the macro in a different context, it is recommended that you yourself verify that the transfer is successful. If the transfer is not successful, then fix the problem and resubmit at least the %CMFTPSND macro and the preceding %CPSTART macro.*

- *For %CPARCRET*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the %CPARCRET macro and the preceding %CPSTART macro.

  If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the message in the SAS log, fix the problem, and resubmit at least the %CPARCRET macro and the preceding %CPSTART macro.

  In either type of failure it is not necessary to delete the PDB to which the data was (or would have been) retrieved, and it is not necessary to delete any data that might have been retrieved to that PDB.

- *For %CPAVAIL*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the %CPAVAIL macro and the preceding %CPSTART macro.

If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the message in the SAS log, fix the problem, and resubmit at least the %CPAVAIL macro and the preceding %CPSTART macro.

In both the cases above, if the resubmission does not result in the completion of the macro's execution with a value of zero for the return code, then remove the new table definition and remove the two new views. Then resubmit at least the %CPAVAIL macro and the preceding %CPSTART macro.

To remove the new table definition, use the %CPDDUTL macro to apply a DELETE TABLE control statement for the new table (the table that you specified by using the OUTABLE= parameter in the %CPAVAIL macro). Here is an example:

```
%CPCAT; cards4;
delete table name=<new-table-name>;
;;;; * semicolons must begin in column 1 ;
%CPCAT( cat=work.temp.temp.source );
%CPDDUTL( entrynam=work.temp.temp.source, list=y );
```

To remove the two new views, use the SAS DATASETS procedure, as in this example:

```
proc datasets lib=admin;
  * The name of the table as specified by OUTABLE= in %CPAVAIL ;
    delete  <new-table-name> / memtype=view;
  * The name of the same table, this time with a suffix of "D" ;
    delete D <new-table-name>D / memtype=view;
    quit;
    run;
```

- *For %CPBATCH*

  Check the SAS log to see which of the other macros failed and then see that macro's directions for troubleshooting. (The %CPBATCH macro runs one or more other macros.) When the problem is fixed, resubmit at least the %CPBATCH macro and the preceding %CPSTART macro.

- *For %CPCAT, %CPHDAY, %CPDUPINT, and %CPTFORM*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the macro for which the error occurred and the preceding %CPSTART macro.

  If the macro completes its execution but writes an error message to the SAS log, then read the message, fix the problem, and resubmit at least the macro in which the error occurred and the preceding %CPSTART macro.

- *For %CPDEACT and %CPFAILIF*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem. Next, if the SAS log from the partial run is available, look at the log in order to see the value of the return code from that run. Typically, before this macro there is a %PUT statement that writes the value of the return code to the SAS log. Here is an example:

  ```
  %PUT return code is
     &macro-variable-name ;
  ```

  In this example, *&macro-variable-name* is the name of the macro variable from the previous macro's _RC= parameter.

  Also, see if the log has any error messages from the previous macro call. Then, based on that information, edit a copy of the job as appropriate for recovery, and resubmit the edited job.

If the macro completes its execution but writes an error message to the SAS log, then read the message and fix the problem. If you want to continue running the job, then look in the SAS log of the failed job for the value of the return code from the previous macro, and edit a copy of the job to add a %LET statement immediately before the macro that produced the error (as shown in the example below). Then resubmit the edited job.

```
%LET name-of-variable-in-_RC-parameter=
     its-value-printed-by-PUT-statement;
```

- *For %CPDBCOPY*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then use operating system commands to delete as much of the *toPDB* as was created; fix the problem that caused the macro to fail; reallocate the *toPDB* if on z/OS; and resubmit at least the %CPDBCOPY macro and the preceding %CPSTART macro.

  If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log; use operating system commands to delete as much of the *toPDB* as was created; fix the problem that caused the macro to fail; reallocate the *toPDB* if on z/OS; and resubmit at least the %CPDBCOPY macro and the preceding %CPSTART macro.

- *For %CPDBPURG and %CPINSPKG*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then restore the PDB from the backup that was made after the most recent process-and-reduce run, fix the problem, and resubmit at least the %CPDBPURG macro and the preceding %CPSTART macro.

  If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log, and based on the message decide whether or not to restore the backup of the PDB that was made after the most recent process-and-reduce run (when in doubt, restore the backup of the PDB). Then fix the problem and resubmit at least the %CPDBPURG macro and the preceding %CPSTART macro.

- *For %CPDDUTL*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), or if the macro completes its execution but writes an error message to the SAS log, then fix the problem. Then do the following.

  - If %CPDDUTL failed because the disk was full or failed while applying an ADD, BUILD, CREATE, DELETE, INSTALL, MAINTAIN, PURGE, SYNCHRONIZE, or UPDATE control statement, then restore the PDB from the backup that you made after the most recent process-and-reduce run, and resubmit at least the %CPDDUTL macro and the preceding %CPSTART macro.
  - If %CPDDUTL failed while applying a COMPARE, END, GENERATE, INCLUDE, LIST, PRINT, QUIT, SAVE, SET, STATUS, STOP, VERIFY, or XREF control statement, then you do not need to restore the PDB from backup. However, you do need to revise the file or catalog entry of control statements that %CPDDUTL was applying: take out the control statements that were already applied before the problem occurred, and if the value of the active table was defined before the problem occurred, then put in a SET TABLE control statement in order to specify that table as the value of the active table. Then resubmit at least the %CPDDUTL macro and the preceding %CPSTART macro.

- *For %CPDUPCHK and %CPDUPDSN*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit the whole job.

  If the macro completes its execution but writes an error message to the SAS log, then read the message, fix the problem, and resubmit the whole job.

- *For %CPDUPUPD*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem, restore the PDB from the backup that was made after the most recent process-and-reduce run, and resubmit the whole job.

  If the macro completes its execution but writes an error message to the SAS log, then read the message, restore the PDB from the backup that was made after the most recent process-and-reduce run, fix the problem, and resubmit the whole job.

- *For %CPEDIT, %CPLVLTRM, %CPRENAME, and %CPUNCVT*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then restore the PDB from the backup that was made after the most recent process-and-reduce run, fix the problem, and resubmit at least the macro that caused the error and the preceding %CPSTART macro.

  If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log, restore the PDB from the backup that was made after the most recent process-and-reduce run, fix the problem, and resubmit at least the macro that caused the error and the preceding %CPSTART macro.

- *For %CPLOGRC*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and check the log for all messages that relate to the macros that ran before this macro. If all those macros ran successfully, then make a copy of the job, delete the macros that executed successfully (but do not delete the preceding %CPSTART macro or the %CPLOGRC macro call that includes START as the value for the *opcode* parameter), and submit the edited copy of the job.

  If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a value other than 0 or 4 (in this macro, 4 does not indicate a problem), then read the error message in the SAS log and fix the problem. (If any macros did not complete their execution successfully, then follow the troubleshooting instructions for those macros.) Then make a copy of the job, delete macro calls that executed successfully (but do not delete the preceding %CPSTART macro or the START call to %CPLOGRC), and submit the edited copy of the job.

- *For %CPSEP*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the statement that has a call to %CPSEP and the preceding %CPSTART macro.

- *For %CSATR2DD, %CSCSIFMT, and %CSSNMSCH*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the macro in which the error occurred and the preceding %CPSTART macro.

  If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log, fix the problem, and resubmit at least the macro in which the error occurred and the preceding %CPSTART macro. If problems still exist, then contact SAS

Technical Support. Be prepared to provide a copy of the failed job and a copy of the input SAS data set(s) or file(s).

- *For %CPSRCRPT*

  If the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), or if the macro completes its execution with a nonzero value of the macro variable that is specified in the _RC= parameter, then troubleshooting and recovery depend on the SAS program that is specified in the *rptname* parameter.

- For all other SAS IT Resource Management macros (*%CMEXTDET, %CPACCFMT, %CPACCUTL, %CPC2RATE, %CPDBKILL, %CPEISSUM, %CPENTCPY, %CPEXCEPT, %CPHTREE, %CPIDTOPN, %CPMANRPT, %CPPDBOPT, %CPPKGCOL, %CPRAWLST, %CPRPTPKG, %CPRPT2DD, %CPDD2RPT, %CPSETHAX, %CPWEBINI*), if the macro fails before completion (because of a power failure, out-of-time condition, or similar problem), then fix the problem and resubmit at least the macro in which the error occurred and the preceding %CPSTART macro.

  If the macro completes its execution and the macro variable that is specified in the _RC= parameter has a nonzero value, then read the error message in the SAS log, fix the problem, and resubmit at least the macro in which the error occurred and the preceding %CPSTART macro.

Guidelines for Allocating Space in the PDB

DETAIL

Ascertain the amount of space that is used in the DETAIL library for one day's data. Multiply this amount by the age limit that is typical in the PDB for the detail level of tables that have a Kept status of YES. Then multiply that value by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

Backloading data is an exception. If you are backloading data, then remember that incoming data is not aged out of the detail level; only existing data is aged out of the detail level. For example, think about a situation in which all of the following apply:

- the age limit in detail level is 5 days
- the detail level has 5 days of data
- you backload into the detail level 7 days of data from last month.

While you process and reduce that backloaded data, the detail level must be able to hold not only 5 days of data but also an additional 7 days of data. Existing data is eligible for aging out, but in this case it is not aged out because the existing data is still from the most recent 5 days. Additionally, the incoming data is not eligible for aging. The next time that you process data, those 7 days from last month will be aged out. This happens because by then they are part of the existing data in the detail level, are eligible for aging out, and are older than the table's age limit for the detail level.

DAY

Ascertain the amount of space that is used in the DAY library for one day's data; multiply by the age limit that is typical in the PDB for the day level of tables that have a Kept status of YES and have at least one statistic that is selected at day level; and multiply by at

least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

At the day level, data that is older than permitted by the table's day-level age limit is not used to update the day level. Thus, backloading data affects the size of the day level only if the backloaded data is within the table's day-level age limit.

WEEK — Ascertain the amount of space that is used in the WEEK library for one week's data; multiply by the age limit that is typical in the PDB for the week level of tables that have a Kept status of YES and have at least one statistic that is selected at week level; and multiply by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

At the week level, data that is older than permitted by the table's week-level age limit is not used to update the week level. Thus, backloading data affects the size of the week level only if the backloaded data is within the table's week-level age limit.

MONTH — Ascertain the amount of space that is used in the MONTH library for one month's data; multiply by the age limit that is typical in the PDB for the month level of tables that have a Kept status of YES and have at least one statistic that is selected at month level; and multiply by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

At the month level, data that is older than permitted by the table's month-level age limit is not used to update the month level. Thus, backloading data affects the size of the month level only if the backloaded data is within the table's month-level age limit.

YEAR — Ascertain the amount of space that is used in the YEAR library for one year's data; multiply by the age limit that is typical in the PDB for the year level of tables that have a Kept status of YES and have at least one statistic that is selected at year level; and multiply by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

At the year level, data that is older than permitted by the table's year-level age limit is not used to update the year level. Thus, backloading data affects the size of the year level only if the backloaded data is within the table's year-level age limit.

DICTLIB — Ascertain the amount of space that is used in the DICTLIB library and multiply by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

ADMIN — Ascertain the amount of space that is used in the ADMIN library and multiply by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

COLLECT

- On z/OS, after you have been running for at least the number of days that are specified in the IMACSPIN library, ascertain the amount of space that is used and multiply by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.) If you increase the number of days that are specified in the IMACSPIN library, then you should make a corresponding change in the size of the

COLLECT library. That is, if the number of days increases by 20 percent, then increase the size by 20 percent.

- For UNIX and Windows environments, ascertain the amount of space that is used in the DETAIL library by one day's worth of data, multiply by the number of days' worth of data that is being processed in each run, and multiply by at least 2.2 in order to include a safety factor.

PDBWORK Submit these statements from the SAS Program Editor window in order to ascertain the amount of space that is used by the largest data set at detail level:

```
libname detail 'pdb_name.detail'
   access-parameter=access-value ;
proc contents data=detail._all_ ;
run;
```

In this example, *pdb_name* represents the name of the PDB, and the value of *access-parameter=access-value* should be DISP=SHR for z/OS and ACCESS=READONLY for UNIX or Windows.

In the output from this code, the size of the data sets is listed in the column Total Blocks Used, which is in the section Engine/Host Dependent Information. Ascertain the largest amount of space in that column, multiply by 4, and then multiply by at least 2.2 in order to include a safety factor. (On z/OS, at least half of the space should be a primary allocation.)

Note: To estimate the size of a new production PDB, use an operating system utility such as

- z/OS: ISPF or SPF
- UNIX: du -s
- Windows: Windows Explorer

to check for the amount of space used by each library of your PDB. This amount indicates the amount needed for one unit of data (where a unit is a day for the DETAIL library and the DAY library, a week for the WEEK library, a month for the MONTH library, and a year for the YEAR library). Extrapolate for the length of time (age limit) that you selected for each level.

The amount of data at the day, week, month, and year summary levels is a function of the number of unique combinations of class variables values that are present in your data. △

Note: The amount of space used at week, month, and year levels does not show any appreciable increase until data for another week, month, or year are processed. The data currently stored in the week level, for example, represents a summarization of the whole week and, as such, will be modified by updates to the data during the week, not by additions. △

Guidelines for Revising Space Allocation in the PDB

- For UNIX or Windows: One solution is to arrange for an increase of your quota on the drive where your PDB resides or to move other directories and files from that directory so that the PDB has more room.

Another solution is to use your operating environment commands to copy the PDB directory (the one with the name of the PDB) and everything under it to a drive where it will have more room, and delete the PDB and everything under it from the original drive. Or use a batch job with %CPSTART and %CPDBCOPY macros in order to copy the PDB.

In the case of the second solution, next, before you do anything else in interactive mode, change the PDB location in your list of known PDBs and check the location of the archive. If it is under the PDB, then update the location. For more information, follow this path: **Manage PDBs ▶ Properties ▶ Help**.

Similarly, before you do anything else in batch mode, check all batch jobs for mention of that PDB (and archive, if it is under the PDB), and change the locations that are specified for them.

Then notify other users of the PDB, because they will want to change the location of the PDB in their list of known PDBs, and they might need to make changes in the remote profile that they associated with the PDB or to make a separate remote profile for the PDB in its new location.

- For z/OS: For each library that has insufficient space, rename the original library, allocate a new library that has the name that the original library used to have and the same blocking size as the original library, use the SAS DATASETS procedure (PROC DATASETS) to copy the contents of the original library to the new library, and then delete the original library.

  For more information about PROC DATASETS, see the *Procedures Guide* for your current release of SAS.

Guidelines for Allocating Space in the Archive

Ascertain the amount of space that is used in the DETAIL library for one day's data, and then multiply by 1.1 in order to include a safety factor. Consider this to be a reasonable estimate for the size of each archive library.

If archiving is enabled, then one archive library will be generated by each process run.

Thus, the size of the archive will be the size of one archive library, multiplied by the number of process runs. Note that the size of the archive increases indefinitely. If you want to control the size of the archive, then you can archive directly to tape instead of to disk. If you instead archive to tape in two steps (first to disk, and then from disk to tape), then when you want to retrieve data from the archive you must restore the relevant archive libraries to the locations from which they were copied, because the archive history records them as residing there and thus the %CPARCRET macro will look for them there.

Guidelines for Revising Space Allocations in the Archive

The %CPARCRET macro looks in the PDB's archive history for the location of the data that is to be retrieved. The PDB's archive history stores information separately and permanently for each archive library.

Thus, previously created archive libraries must stay where they were created (or be written to tape and then, when you need to retrieve data from them, restored to the location where they were created).

New archive libraries can go to another location. To direct them to another location, change the archive options in the PDB properties.

APPENDIX

2

Archiving Data

Archive Overview

To archive data that is being processed into the detail level of the PDB, you must do two things:

1. In the PDB properties, set archiving options. By default, the archive is "under" the PDB.
2. For each table whose data you want to archive, turn on archiving in the table properties. By default, the tables are not archived.

When you archive data, the archive library contains data sets for the metadata, catalogs for each of the tables, and data for each of the tables. Data that has been archived can be retrieved into a separate PDB to assist you in reviewing or investigating a problem in the data or to validate the data for a specific problem. The archive utility should not be used as a replacement for your regular system backup of the PDB. (The archive utility handles data only for the detail level. A backup handles data for all levels of the PDB.)

One archive SAS data library is created for each process run if, and only if, archive options have been turned on for at least one of the tables in the process run.

Archiving support is available for any collectors that are supported by the %CMPROCES or %CPPROCES macros, as well as any collectors that are supported by COLLECTR=GENERIC on the %CSPROCES or %CWPROCES macros. (Archiving support is not available for the following collectors: ACCTON, HP-PCS, HP-MWA, HP-OVPA, HP-OV, HP-OV-C, HP-VPPA, PROBEX, SPECTRUM, SUNETMGR, and TRAKKER.)

- You can archive data when you process data, either interactively or in batch mode. Before you process the data you must (1) set the archive option (in the table's properties) for each table that you want to archive in a PDB and (2) set (in the PDB's properties) the archive options for the PDB whose data you want to archive.
- You can retrieve data by using the %CPARCRET macro.

To set archive options for a PDB, use one of the following methods:

- In batch mode, use the %CPPDBOPT macro with the ARCDEV=, ARCENGIN=, ARCPARM=, and ARCPATH= parameters (see "%CPPDBOPT" on page 135).
- From the Administration tab in the SAS IT Resource Management GUI for UNIX or Windows, follow this menu path in order to set PDB options for the active PDB: **Manage PDBs ▶ Properties ▶ Archive**.

- From the SAS IT Resource Management GUI for z/OS, follow this menu path in order to set PDB options for the active PDB: **PDB Admin ▶ Set Active PDB Options**.

  For information on how to complete the Set Active PDB Options window, select **`Help`** in the window.

To activate archiving for a table in a PDB, use one of the following methods in order to set the Archive option (ARCHIVE=YES) for each table that you want to archive:

- In batch mode, use the UPDATE TABLE control statement with the %CPDDUTL macro (see "UPDATE TABLE" on page 639) and specify the ARCHIVE= parameter with a value of YES.
- From the Administration tab in the SAS IT Resource Management GUI for UNIX or Windows, follow these steps:
  1. Select **`Manage Tables`**.
  2. From the list, right-click on the tables that you want to archive.
  3. From the pop-up menu, select **Properties ▶ Advanced**.
  4. From the list of options, select **Archive: Yes**.
- From the SAS IT Resource Management GUI for z/OS, follow these steps:
  1. Select this menu path: **PDB Admin ▶ Config Active PDB Dictionary**.
  2. From the list, select the tables that you want to archive.
  3. Select this menu path: **ItemActions ▶ Edit Definition**.
  4. From the list of options, select **Archive: Yes**.

To retrieve data, use the %CPARCRET macro and use the ARCPDB= parameter to specify the PDB from which the data was archived (see "%CPARCRET" on page 61). The data will be restored to the active PDB.

Note: If you delete a PDB on z/OS and you previously archived data from that PDB to a tape, then you must manually delete the archive libraries in order to avoid future name space problems. (See your site administrator for more information about uncataloging tapes.) △

APPENDIX

3

Error Checking

Error Checking with Return Codes

Macros That Check Return Codes

SAS IT Resource Management provides the following macros that check return codes:

- %CPDEACT
- %CPFAILIF
- %CPLOGRC.

For more information, see "%CPFAILIF" on page 120, "%CPLOGRC" on page 128, and "%CPDEACT" on page 84.

Option for Error Checking

To cause SAS to terminate *immediately* on encountering certain severe errors, include the following statement in the SAS program:

```
options ERRORABEND ERRORCHECK=STRICT;
```

The immediate termination of the SAS job upon detection of a severe error is an additional safeguard that prevents the SAS job from proceeding after an error occurs. This might or might not be desirable for the application.

Note: In SAS IT Resource Management, ERRORABEND is turned on during the process step, reduce step, and other steps that affect the integrity of the data in the tables.

APPENDIX

4

Working with Duplicate Data

Duplicate-Data Checking

Note: This information on duplicate-data checking applies to the %CMPROCES macro and %CPPROCES macro.

For information on duplicate-data checking with the %CSPROCES macro and %CWPROCES macro, see the EXITSRC= parameter on "%CSPROCES" on page 204 or "%CWPROCES" on page 220, and see exit point PROC180 in the section "Using Process Exits" in the chapter "Administration: Extensions to SAS IT Resource Management in the *SAS IT Resource Management User's Guide*.

Duplicate-Data Checking Overview

SAS IT Resource Management provides a set of macros that enable you to control whether duplicate data is processed into the PDB by the %CMPROCES or %CPPROCES macro. In this context, duplicate data is defined as data whose datetime stamp is within a range of data that has already been processed into the PDB for that machine or system.

Each of the duplicate-data-checking macros performs a specific task, and together they set up and manage duplicate-data checking.

- The main purpose of the macros is to check your data and to prevent duplicate data from being processed into the PDB.
- However, sometimes it is necessary to process data in a datetime range for which that machine or system's data was already processed. (For example, you may need to process data into a table that you did not use earlier, or to process data into a table that you accidentally deleted.) You can specify to the macros that the data is to be accepted even though it appears to be duplicate data.

This appendix describes how to set up duplicate-data checking with %CMPROCES or %CPPROCES, for your type of data. First, there is a continuation of the overview, including a brief introduction to the components in duplicate-data checking. Then, for each type of data, there is a separate set of instructions for implementing duplicate-data checking.

What Is Duplicate-Data Checking?

As raw data is being read, one of the duplicate-data-checking macros reviews the datetime information in each record and stores the information in a SAS data set called a *temporary* control data set. Later, by using *intermediate* control data sets, another macro merges the information in the temporary control data set into one or more SAS data sets that are called *permanent* control data sets.

When additional data is processed into the PDB, the timestamps of the incoming data are compared with the datetime information in the permanent control data sets in order to determine whether the new data has already been processed. If it has, the duplicate data is handled in the way that you specify.

A duplicate-data report is printed in the SAS log after the data is read. The report describes how many records were read for each machine or system and how many duplicates were found, if any.

Note: The first time that you use the macros, the permanent control data sets have not been built, so %CPDUPCHK cannot check the input records. Your data is not checked for duplicates or rejected (but the permanent control data sets will be created and the datetime information for this data will be saved to them). Also, the duplicate-data report contains only a limited amount of information about your data. △

Control Data Sets for Duplicate-Data Checking

Control data sets (temporary, intermediate, and permanent) are the basis by which duplicate data is rejected. Therefore, it is important that you understand how the control data sets are created, used, and updated as well as where they are stored.

- Temporary: When the input data is processed, the %CPDUPCHK macro creates a temporary control data set, WORK._DUPCNTL, which stores information about the raw data from one or more collectors. Specifically, for each machine or system

that generated data, the temporary control data set stores the datetime ranges and record counts in the raw data. If raw data from more than one data source is processed in the same job, then information is appended to the temporary control data set with each execution of the %CPDUPCHK macro.

If WORK._DUPCNTL exists, then the new data is appended to it. Otherwise, WORK._DUPCNTL is created and the new data is written to that data set.

- Intermediate: When all the data has been read and stored in the temporary control data set, the %CPDUPUPD macro writes the data from the temporary control data set WORK._DUPCNTL, to separate intermediate (one per collector) data sets, which are named ADMIN.sourceDAY, where "source" is a three-character identifier of the data source or data collector.

  If an ADMIN.sourceDAY data set exists, then the data set is used and the new data overwrites the old data; otherwise, the data set is created and the new data is written to that data set.

- Permanent: The data in these intermediate control data sets is then merged by %CPDUPUPD into the corresponding permanent control data sets, which are named ADMIN.sourceCNTRL. The permanent control data sets are stored and maintained in the ADMIN library of the PDB. There is one permanent control data set for each collector, and it is named ADMIN.sourceCNTRL. Each data set contains information about that collector's machines or systems, datetime ranges, and record counts.

  If an ADMIN.sourceCNTRL data set exists, the new data is merged into it; otherwise, the data set is created and the new data is written to that data set.

The Content of Permanent Control Data Sets

The content of the permanent control data sets is based on intervals and ranges in your data. When you specify the INT= parameter on %CPDUPCHK, you define the maximum gap allowable between records in the same range. If the gap between datetimes for two consecutive records exceeds this value, then a new range is created. When you specify the RANGE= parameter on %CPDUPCHK, you define the maximum number of ranges that are allowed to be in the raw data during the current execution of %CPPROCES or %CMPROCES. (If the raw data exceeds the maximum number of ranges, then processing stops and you receive an error message.) Thus, the RANGE= and INT= parameters work together to flag datetime gaps.

- If your data is continuous and does not have any datetime gaps that exceed the value of the INT= parameter, then your data always updates the same range. In this case the permanent control data sets contain one range for each unique value of the variable that is specified by the IDVAR= parameter. (The values of that variable typically are the machine or system names from which the raw data originated.)
- If your data is not continuous, then the permanent control data sets contain multiple ranges for each unique value of the variable that is specified by the IDVAR= parameter. Each range is prefixed with a value of the variable that is specified by the IDVAR= parameter.

Note: A range is deleted when the end-of-range datetime value is older than the number of weeks that are specified on the KEEP= parameter on %CPDUPCHK. However, if your data is continuous, then you have only one range. Your control information is never aged out, because the end-of-range datetime value is constantly extended by new datetime information. △

The Duplicate-Data-Checking Macros

The duplicate-data-checking macros and short descriptions of the tasks that they perform are listed below. How and when you should use these macros depends on the type of data that you are using. Before you begin using these macros, review the section below titled "Implementing Duplicate-Data-Checking Macros."

- %CPDUPINT loads macro definitions that are used by the other duplicate-data-checking macros.
- %CPDUPDSN generates the name (WORK._DUPCNTL) of the temporary SAS data set that will contain datetime ranges for the data that is being processed into the active PDB.
- %CPDUPCHK checks for duplicate data by examining timestamps on data that is being read by the staging code. It also writes to the temporary control data set.
- %CPDUPUPD updates the permanent control data sets with information from a temporary control data set (by way of the intermediate control data sets).

Implementing Duplicate-Data-Checking Macros

First, set up processing without duplicate-data checking.

Then, implement duplicate-data checking. (Detailed implementation instructions follow the implementation overview.)

Implementation overview for duplicate-data checking

- If you *do* want the process task to do duplicate-data checking, in the call to %CMPROCES or %CPPROCES, set DUPMODE=FORCE, DISCARD, or TERMINATE.
  - If the COLLECTR= parameter on the %CMPROCES or %CPPROCES call *is not* set to GENERIC and is not set to a collector name that is used as a tag (along with tool name) on collector-support entities that will be or have been packaged and installed (by using %CPPKGCOL and CPINSPKG) for use with %CPPROCES, then
    - The %CMPROCES or %CPPROCES macro arranges for the %CPDUPINT, %CPDUPDSN, and %CPDUPCHK macro calls to be in the collector-specific staging code.
    - If the call to the %CMPROCES or %CPPROCES macro has DUPMODE=FORCE and/or the call to the %CPDUPCHK macro has FORCE=YES, "duplicate" data is allowed into the PDB. (The "duplicate" data should be data that may appear to the duplicate-data-checking macros to be duplicate, but is not actually duplicate. For example, the "duplicate" data was processed into the PDB before but was accidentally deleted. Or the "duplicate" data is being processed into a table that was not in use in the PDB when the data was processed earlier.)

      If the call to the %CMPROCES or %CPPROCES macro has DUPMODE=TERMINATE and/or the call to the %CPDUPCHK macro has TERM=YES, processing will terminate if duplicate data is encountered.

      If the call to the %CMPROCES or %CPPROCES macro has DUPMODE=DISCARD and the call to the %CPDUPCHK macro does not have FORCE=YES and does not have TERMINATE=YES, the duplicate data is discarded.

- In the non-collector-specific staging code, %CMPROCES or %CPPROCES calls the %CPDUPDSN macro before the merge of incoming and existing data.

- If the COLLECTR= parameter on the %CMPROCES or %CPPROCES call *is* set to GENERIC or is set to a collector name that is used as a tag (along with tool name) on collector-support entities that will be or have been packaged and installed (by using %CPPKGCOL and %CPINSPKG) for use with %CPPROCES, then
  - you must place the calls to %CPDUPINT, %CPDUPDSN, and %CPDUPCHK macros in the collector-specific staging code.
  - you must use set the values of the FORCE= and TERM= macros in the call to %CPDUPCHK.
  - In the non-collector-specific staging code, %CMPROCES or %CPPROCES calls the %CPDUPDSN macro before the merge of incoming and existing data.

- If you *do not* want the process task to do duplicate-data checking, in the call to %CMPROCES or %CPPROCES, set DUPMODE=INACTIVE. (And if COLLECTR=GENERIC or if a collector name that is used as a tag (along with tool name) on collector-support entities that will be or have been packaged and installed (by using %CPPKGCOL and %CPINSPKG) for use with %CPPROCES, check that the collector-specific staging code does not call the duplicate-data-checking macros.)

For detailed implementation instructions, see the appropriate one of these four cases, depending on how you implemented processing:

1. Using supplied table definitions and supplied MXG-based staging code (see "Supplied Table Definitions and Supplied MXG-Based Staging Code" on page 676)
2. Using supplied or user-generated table definitions and supplied Non-MXG-Based staging code (see "Supplied or User-Generated Table Definitions and Supplied Non-MXG-Based Staging Code" on page 679)
3. Using user-generated table definitions and supplied MXG-based staging code (see "User-Generated Table Definitions and MXG-Based Staging Code" on page 683)
4. Using user-generated table definitions and user-generated non-MXG-based staging code (see "User-Generated Table Definitions and User-Generated Non-MXG-Based Staging Code" on page 691).

Note: For supplied staging code, you can determine whether it is MXG-based by looking at any %CxPROCES macro in the SAS IT Resource Management macro reference documentation. (%CxPROCES means %CMPROCES, %CPPROCES, %CSPROCES, or %CPPROCES.) Select the COLLECTR= parameter in the macro's syntax. The documentation for the COLLECTR= parameter has a Collector/Toolname table. In the table, find the appropriate row for your data source.

- If the row's value for TOOLNM= is MXG, then your data source uses collector-specific staging code that is MXG-based.
- If the row's value for TOOLNM= is not MXG, then your data source uses collector-specific staging code that is non-MXG-based.

△

Note: Accepting "duplicate" data:
The duplicate-data-checking macros are designed to prevent the same data from being processed into the PDB twice. However, there may be times when you need to

process some data that is in a datetime range for which the permanent control data sets have already recorded machine or system data. (For example, you may need to process data into one or more tables that you did not use earlier, or to process data into one or more tables that you accidentally purged or deleted.) Remember to restore the DUPMODE=FORCE or FORCE=YES setting to its original value after you finish the process task. △

Supplied Table Definitions and Supplied MXG-Based Staging Code

Use these instructions if you

- process data by calling %CMPROCES or %CPPROCES,
- on the call to %CMPROCES or %CPPROCES, set the COLLECTR= parameter to a value that is not GENERIC,
- on the call to %CMPROCES or %CPPROCES, set the TOOLNM= parameter to MXG,

which is the case for the following values of the COLLECTR= parameter (and process macro): DCOLLECT (%CM), EREP (%CM), IMF (%CM), NTSMF (%CM), SMF (%CM or %CP), TMON2CIC (%CM), TMONCIC8 (%CM), TMONCICS (%CM), TMONDB2 (%CM), TMS (%CM), TPF (%CM), and VMMON (%CM).

Here is an overview of the preparation for duplicate-data checking for this case:

1. On your call to the %CMPROCES or %CPPROCES macro, set the DUPMODE= parameter to FORCE, DISCARD, or TERMINATE.
2. Create a member or file that contains a call to the %CPDUPCHK macro.
3. Review that member or file and modify it, if necessary.
4. The %CMPROCES or %CPPROCES macro will automatically insert a call to the %CPDUPINT macro in the collector-specific staging code.
5. The %CMPROCES or %CPPROCES macro will automatically insert a call to the %CPDUPDSN macro in the collector-specific staging code.
6. The %CMPROCES or %CPPROCES macro will automatically include into the collector-specific staging code the member or file that contains the call to %CPDUPCHK.
7. In the non-collector-specific staging code, the %CMPROCES or %CPPROCES macro will automatically call the %CPDUPUPD macro.

The following sections describe, in detail, the steps that you perform.

Set the DUPMODE= Parameter

On the %CMPROCES or %CPPROCES macro call, set the DUPMODE= parameter to TERMINATE, DISCARD, or FORCE. Each of these values indicates that you do want to use duplicate-data checking. The particular value that you specify indicates how you want duplicate data to be handled.

- To cause processing to stop if duplicate data is encountered, set DUPMODE=TERMINATE.
- To cause processing to continue but duplicate data to be rejected if duplicate data is encountered, specify DUPMODE=DISCARD.
- To cause processing to continue and duplicate data to be accepted if duplicate data is encountered, specify DUPMODE=FORCE. (Use this setting only when loading

data that is not duplicate, even though it may be in the same datetime range and for the same machine or system as previously loaded data.)

Note: Do not specify DUPMODE=INACTIVE. (Also, do not omit specifying the DUPMODE= parameter, because DUPMODE= defaults to INACTIVE.) If DUPMODE= is INACTIVE, duplicate-data checking is not used. △

Create the Member or File

The default call to the %CPDUPCHK macro is contained in a different member or file on each operating system.

- On UNIX, copy the **`dup_smf.sas`** file from the SAS IT Resource Management **`sasmisc`** directory to the *MXG.USERID.SOURCLIB* directory. Keep the same filename.
- On Windows, copy the **`dup_smf.sas`** file from the SAS IT Resource Management **`misc`** directory to the *MXG.USERID.SOURCLIB* directory. Keep the same filename.
- On z/OS, in the following table find the collector that you are using and the version of SAS (such as 8.2 or 9.1.3 or later) that you are using. That combination of collector and version has one or more members. Copy that member or those members from the SAS IT Resource Management **`CPMISC`** PDS to the *MXG.USERID.SOURCLIB* PDS. Keep the same member name(s).

Table A4.1 Each Collector's CMDUPxxx Member(s), CPDUPxxx Member, or $$DUPxxx Member(s)

COLLECTR=	SAS	Member(s)
DCOLLECT	SAS 8.2	$$DUPDCO
	SAS 9.1.3 or later	CMDUPDCO
EREP	SAS 8.2	$$DUPERP, $$DUPER1, $$DUPER2
	SAS 9.1.3 or later	CMDUPERP, CMDUPER1, CMDUPER2
IMF	SAS 8.2	$$DUPIMF
	SAS 9.1.3 or later	CMDUPIMF
NTSMF	SAS 8.2	$$DUPNTS
	SAS 9.1.3 or later	CMDUPNTS
SMF	SAS 8.2	$$DUPSMF
	SAS 9.1.3 or later	CPDUPSMF
TMONCICS	SAS 8.2	$$DUPTMC
	SAS 9.1.3 or later	CMDUPTMC
TMONCIC8	SAS 8.2	$$DUPTM8
	SAS 9.1.3 or later	CMDUPTM8

COLLECTR=	SAS	Member(s)
TMONDB2	SAS 8.2	$$DUPTMD
	SAS 9.1.3 or later	CMDUPTMD
TMON2CIC	SAS 8.2	$$DUPTM2
	SAS 9.1.3 or later	CMDUPTM2
TMS	SAS 8.2	$$DUPTMS
	SAS 9.1.3 or later	CMDUPTMS
TPF	SAS 8.2	$$DUPTPF
	SAS 9.1.3 or later	CMDUPTPF
VMMON	SAS 8.2	$$DUPVMM
	SAS 9.1.3 or later	CMDUPVMM

Note: If you run SAS IT Resource Management on top of SAS 8.2 or earlier, the member names begin with *$$*. If you run SAS IT Resource Management on top of SAS 9.1.3 or later, the member names begin with *CM* or *CP*.

If you run SAS IT Resource Management on SAS 8.2 and then on SAS 9.1.3 or later, members that were already stored in the MXG source library with the prefix **$$** will continue to be recognized and used. △

Modify the Member or File

The new member (on z/OS) or new file (on UNIX or Windows) contains a call to the %CPDUPCHK macro. The call is similar to this one for SMF:

```
%CPDUPCHK(SOURCE=SMF, IDVAR=SYSTEM, TIMESTMP=SMFTIME,
          ENDFILE=ENDOFSMF, INT=00:29, SYSTEMS=3,
          RANGES=20, KEEP=9);
```

The parameters and values that are used in this example are defined below:

SOURCE=*identifier*
a three-character identifier of the data source or data collector. This must be set to SMF for SMF data.

IDVAR=*variable-name*
identifies the SAS variable that is used by MXG to denote the origin of each SMF record. This must be set to SYSTEM for the SMF collector, because that is the name of the variable that is used by MXG.

TIMESTMP=*SAS-timestamp-variable*
identifies the SAS variable that is used by MXG to record the datetime stamp of each SMF record. The variable name must be SMFTIME for the SMF collector, because that is the name of the timestamp variable that is used by MXG.

ENDFILE=*variable-name*

identifies the name of the SAS variable that is used by MXG on the INFILE statement to denote the end-of-file condition for the SMF data. The variable name must be ENDOFSMF for the SMF collector.

INT=*interval*
represents the maximum interval that is permitted between the datetime stamps on any two consecutive SMF records from the same z/OS system. If the interval between the datetime stamp values exceeds the value of this parameter, then a new range is created. (See the RANGES= parameter.) For SMF=, the default value is 29 minutes.

SYSTEMS=*number-of-systems*
represents an estimate of the maximum number of z/OS systems for which your file contains data. For SMF, the default value indicates a maximum of three systems.

RANGES=*number-of-ranges*
represents the maximum number of ranges that can occur during this execution of %CMPROCES or %CPPROCES. A new range is created when the difference between the datetime stamps of two consecutive records exceeds the value of the INT= parameter. This break is referred to as a gap in the data. For SMF, the default value indicates a maximum of 20 ranges. This means that there can be no more than 19 gaps larger than 29 minutes (the value of the INT= parameter).

KEEP=*number-of-weeks*
represents the maximum number of weeks for which you want to retain control data. A range is aged out (removed) when the end-of-range datetime is older than permitted by the value of this parameter. For SMF, the default value is nine weeks.

For a complete description of these and other parameters for this macro, see "%CPDUPCHK" on page 86.

Review and modify your collector's default call as follows:

- Review the values of the INT=, SYSTEMS=, RANGES=, and KEEP= parameters, and change those values to be appropriate for your site.
- Do not use the TERM= parameter or FORCE= parameter. (The default settings are TERM=NO and FORCE=NO.) If the call to %CMPROCES or %CPPROCES has DUPMODE=TERMINATE, that will cause termination to occur if duplicate data is encountered, without requiring the setting of TERM=YES on %CPDUPCHK. Similarly, if the call to %CMPROCES or %CPPROCES has DUPMODE=FORCE, that will enable duplicate data to be processed, without requiring the setting of FORCE=YES on %CPDUPCHK.
- Do not change any of the other parameters or values in your call.

Supplied or User-Generated Table Definitions and Supplied Non-MXG-Based Staging Code

Use these instructions if you

- process data by calling %CMPROCES or %CPPROCES
- on the call to %CMPROCES or %CPPROCES, set the COLLECTR= parameter to a value for supplied collector support that is non-MXG based
- on the call to %CMPROCES or %CPPROCES, set the TOOLNM= parameter to a value that is not MXG,

which is the case for the following values of the COLLECTR= parameter (and process macro): ACCUNX (%CP), ETEWATCH (%CP), NTSMF (%CP), PATROL (%CP),

ROLMPBX (%CP), SAPR3 (%CP), SAR (%CP), WEBLOG (%CP), and GENERIC (if the tool name is CHARDELIM).

Here is an overview of the preparation for duplicate-data checking for this case:

1. On your call to the %CMPROCES or %CPPROCES macro, set the DUPMODE= parameter to FORCE, DISCARD, or TERMINATE.
2. Create an entry that contains a call to the %CPDUPCHK macro.
3. Review that entry and modify it, if necessary.
4. The %CMPROCES or %CPPROCES macro will automatically insert a call to the %CPDUPINT macro in the collector-specific staging code.
5. The %CMPROCES or %CPPROCES macro will automatically insert a call to the %CPDUPDSN macro in the collector-specific staging code.
6. The %CMPROCES or %CPPROCES macro will automatically include into the collector-specific staging code the entry that contains the call to %CPDUPCHK.
7. In the non-collector-specific staging code, the %CMPROCES or %CPPROCES macro will automatically call the %CPDUPUPD macro.

The following sections describe, in detail, the steps that you perform.

Set the DUPMODE= Parameter

On the %CMPROCES or %CPPROCES macro call, set the DUPMODE= parameter to TERMINATE, DISCARD, or FORCE. Each of these values indicates that you do want to use duplicate-data checking. The particular value that you specify indicates how you intend to set the FORCE= and TERM= parameters on the call the %CPDUPCHK macro.

- To cause processing to stop if duplicate data is encountered, set DUPMODE=TERMINATE.
- To cause processing to continue but duplicate data to be rejected if duplicate data is encountered, specify DUPMODE=DISCARD.
- To cause processing to continue and duplicate data to be accepted if duplicate data is encountered, specify DUPMODE=FORCE. (Use this setting only when loading data that is not duplicate, even though it may be in the same datetime range and for the same machine or system as previously loaded data.)

Note: Do not specify DUPMODE=INACTIVE. (Also, do not omit specifying the DUPMODE= parameter, because DUPMODE= defaults to INACTIVE.) If DUPMODE= is INACTIVE, duplicate-data checking is not used. △

Create the Entry

The default call to the %CPDUPCHK macro is contained in the SAS IT Resource Management library PGMLIB. The call is in a catalog entry named **PGMLIB.*collector_name*.CPDUPCHK.SOURCE**.

The %CMPROCES or %CPPROCES macro will look for the call in the following location:

Table A4.2 Source Locations for Each Collector

COLLECTR=	%CPDUPCHK Location
ACCUNX	ADMIN.ACCUNX.CPDUPCHK.SOURCE
ETEWATCH	ADMIN.ETEWATCH.CPDUPCHK.SOURCE
HP-PCS	ADMIN.HPOVPA.CPDUPCHK.SOURCE

COLLECTR=	%CPDUPCHK Location
NETCONV	ADMIN.NETCONV.CPDUPCHK.SOURCE
NTSMF	ADMIN.NTSMF.CPDUPCHK.SOURCE
PATROL	ADMIN.PATROL.CPDUPCHK.SOURCE
ROLMPBX	ADMIN.ROLMPBX.CPDUPCHK.SOURCE
SAPR3	ADMIN.SAPR3.CPDUPCHK.SOURCE
SAR	ADMIN.SAR.CPDUPCHK.SOURCE
SITESCOP	ADMIN.SITESCOP.CPDUPCHK.SOURCE
VISULIZR	ADMIN.VISULIZR.CPDUPCHK.SOURCE
WEBLOG	ADMIN.WEBLOG.CPDUPCHK.SOURCE
GENERIC	ADMIN.GENERIC.CPDUPCHK.SOURCE

Copy the default entry in PGMLIB to the appropriate location in ADMIN:

1. Start the SAS IT Resource Management GUI.
2. Activate the PDB where the collector's data is to be stored. Use WRITE (SHR) mode.
3. Submit the following code from the SAS Program Editor window:

```
PROC CATALOG CAT=PGMLIB.collector_name;
     COPY OUT=ADMIN.collector_name;
     SELECT CPDUPCHK.SOURCE;
     RUN;
     QUIT;
```

   where *collector_name* is the value of the COLLECTR= parameter in the table above.

Note: If %CMPROCES or %CPPROCES is called and there is no such entry in ADMIN, the process macro copies the default entry that is in PGMLIB into the appropriate location in ADMIN and prints a large warning in the SAS log that notifies you of this action. △

Modify the Entry

The new catalog entry contains a call to the %CPDUPCHK macro. To see the call, issue the SAS command

```
NOTE ADMIN.collector_name.CPDUPCHK.SOURCE
```

from a SAS command line (on z/OS) or from the command field on the SAS GUI (on Windows) or SAS Toolbar (on UNIX).

The command opens a Notepad window where the contents of the source entry are displayed. For example, this is the call to %CPDUPCHK for SAP R/3 data:

```
%CPDUPCHK( SOURCE=SAP,
           IDVAR=SYSHOST,
           TIMESTMP=DATETIME,
           ENDFILE=DUPEND,
           INT=00:20,
           RANGES=10,
```

```
SYSTEMS=10,
KEEP=52 );
```

The parameters and values that are used in the above example are defined below.

SOURCE=*identifier*
is a three-character identifier of the data source or data collector. This must be set to SAP for the SAP R/3 data.

IDVAR=*variable-name*
identifies the SAS variable that is used to denote the origin of each SAP R/3 record. The variable name must be SYSHOST for the SAP R/3 collector and must not be changed.

TIMESTMP=*SAS-timestamp-variable*
identifies the SAS variable that is used to record the datetime stamp of each SAP R/3 record. The variable name must be DATETIME for the SAP R/3 collector and must not be changed.

ENDFILE=*variable-name*
identifies the name of the SAS variable that is used on the INFILE statement to denote the end-of-file condition for the SAP R/3 data. The variable name must be DUPEND for the SAP R/3 collector and must not be changed.

INT=*interval*
represents the maximum interval that is permitted between the datetime stamps on any two consecutive SAP R/3 records from the same machine/host combination. If the interval between the timestamp values exceeds the value of this parameter, then a new time range is created. (See the RANGE= parameter.) In this example, the specified value is 20 minutes.

RANGES=*number-of-ranges*
represents the maximum number of ranges that can occur during this execution of %CMPROCES or %CPPROCES. A new range is created when the difference between the datetime stamps of two consecutive records exceeds the value of the INT= parameter. This break is referred to as a gap in the data. For SAP R/3, the default value indicates a maximum of 10 ranges. This means there can be no more than 9 gaps greater than 20 minutes (the value of the INT= parameter).

SYSTEMS=*number-of-systems*
represents an estimate of the maximum number of machine/host combinations for which you expect to process data. The default value indicates a maximum of 10 systems.

KEEP=*number-of-weeks*
represents the maximum number of weeks for which you want to retain control data. A range is aged out (removed) when the end-of-range datetime is older than permitted by the value of this parameter. For SAP R/3 data, the default value is 52 weeks.

For a complete description of these and other parameters for this macro, see "%CPDUPCHK" on page 86.

Review and modify your collector's default call as follows:

- Review the values of the INT=, SYSTEMS=, RANGES=, and KEEP= parameters, and change those values to be appropriate for your site.
- Do not use the TERM= parameter or FORCE= parameter. (The default settings are TERM=NO and FORCE=NO.) If the call to %CMPROCES or %CPPROCES has DUPMODE=TERMINATE, that will cause termination to occur if duplicate data is encountered, without requiring the setting of TERM=YES on

%CPDUPCHK. Similarly, if the call to %CMPROCES or %CPPROCES has DUPMODE=FORCE, that will enable "duplicate" data to be processed, without requiring the setting of FORCE=YES on %CPDUPCHK.

- Do not change any of the other parameters or values in your call.
- When you are finished, press the END key or issue the END command from the command line, to save the change and exit from the Notepad window.

User-Generated Table Definitions and MXG-Based Staging Code

Note: The example that is used below is for illustrative purposes only. The example is based on modifying the MXG code that processes DCOLLECT data, as if you needed to do that. (For DCOLLECT data, SAS IT Resource Management has already done that. Thus, in actual use, for DCOLLECT data you would follow the instructions in the section "Supplied Table Definitions and Supplied MXG-Based Staging Code," and no MXG customization would be necessary.) △

Use these instructions if you

- process data by calling %CMPROCES or %CPPROCES)
- on the call to %CMPROCES or %CPPROCES, set the COLLECTR= parameter to GENERIC
- on the call to %CMPROCES or %CPPROCES, use MXG code to stage the data that will be processed into the user-generated tables.

For information on using the Generic Collector Facility with MXG-based staging code, see the chapter "Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

Here is an overview of the preparation for duplicate-data checking in this case:

1. On your call to the %CMPROCES or %CPPROCES macro, set the DUPMODE= parameter to FORCE, DISCARD, or TERMINATE.
2. Create a member or file for a call to the %CPDUPCHK macro.
3. In that member or file, add the %CPDUPCHK call, and modify the call as necessary.
4. Insert a call to the %CPDUPINT macro in the MXG code.
5. Insert a call to the %CPDUPDSN macro in the MXG code.
6. Include into the MXG code the member or file that contains the call to the %CPDUPCHK macro.
7. In the non-collector-specific staging code, the %CMPROCES or %CPPROCES macro will automatically call the %CPDUPUPD macro.

The following sections describe, in detail, the steps that you do.

Set the DUPMODE= Parameter

On the %CMPROCES or %CPPROCES macro call, set the DUPMODE= parameter to TERMINATE, DISCARD, or FORCE. Each of these values indicates that you do want to use duplicate-data checking. The particular value that you specify indicates how you intend to set the FORCE= and TERM= parameters on the call to the %CPDUPCHK macro.

- To indicate that you intend to stop if duplicate data is encountered, set DUPMODE=TERMINATE.
- To indicate that you intend to cause processing to continue but duplicate data to be rejected if duplicate data is encountered, specify DUPMODE=DISCARD.

- To indicate that you intend to cause processing to continue and duplicate data to be accepted if duplicate data is encountered, specify DUPMODE=FORCE. (Use this setting only when loading data that is not duplicate, even though it may be in the same datetime range and for the same machine or system as previously loaded data.)

Note: Do not specify DUPMODE=INACTIVE. (Also, do not omit specifying the DUPMODE= parameter, because DUPMODE= defaults to INACTIVE.)

If DUPMODE= is INACTIVE, your calls to the %CPDUPINT, %CPDUPDSN, and %CPDUPCHK macros will run, but %CMPROCES or %CPPROCES will not call the %CPDUPUPD macro. △

Create the Member or File

On the host where you will process the data, if there is no member or file in MXG.USERID.SOURCLIB with the following name, create one. Keep the same name.

- On z/OS, member **$$GENxxx**.
- On UNIX, file **gen_xxx.sas**.
- On Windows, file **gen_xxx.sas**.

The xxx is a three-character mnemonic that identifies your data source or data collector. You can choose any mnemonic as long as it is not one of the mnemonics on the list of values for the SOURCE= parameter in "%CPDUPCHK" on page 86.

For example, for DCOLLECT data, you could use DCO.

Modify That Member or File

In this member or file, add a call to %CPDUPCHK.

For example, here is one for DCOLLECT:

```
%CPDUPCHK(SOURCE=DCO,IDVAR=DCUSYSID,TIMESTMP=DCUTMSTP,
          ENDFILE=EOF,INT=25:00,SYSTEMS=2,
          RANGES=3,KEEP=9,
          FORCE=NO,TERM=NO);
```

The parameters and values that are used in this example are defined below:

SOURCE=*identifier*
: specifies a three-character identifier of the data source or data collector. In this example, the value that is used is DCO for DCOLLECT data.

IDVAR=*variable-name*
: specifies the name of the SAS variable that is used by MXG to denote the origin of each record. This example uses the value DCUSYSID, because that is the variable name that is used in VMACDCOL for DCOLLECT data.

TIMESTMP=*timestamp-variable-name*
: specifies the name of the SAS variable that is used by MXG that contains the timestamp of each DCOLLECT record. The variable name must be set to DCUTMSTP for DCOLLECT data, because that is the name used in VMACDCOL. Note that every DCOLLECT record will have the same timestamp because it represents the time when the IDCAMS DCOLLECT facility actually output the records. As shown in this example, you should change the value of the INT= parameter to accommodate this difference.

ENDFILE=*variable-name*

specifies the SAS variable that represents the end-of-file condition for your source data. For DCOLLECT data, this must be set to EOF, because that is the name of the SAS variable that is used by MXG on the INFILE statement for VMACDCOL in order to indicate the end-of-file condition.

INT=*interval*
represents the maximum interval that is allowed between the datetime stamps on any two consecutive DCOLLECT records from the same z/OS system. If the interval between the datetime stamp values exceeds the value of this parameter, then a new range is created (see the RANGES= parameter). In this example, the specified value is 25 hours, because the timestamp for each DCOLLECT record is the same. In effect, this allows the daily DCOLLECT facility to run up to one hour late each day.

SYSTEMS=*number-of-systems*
represents an estimate of the maximum number of z/OS systems for which your file contains data. In this example, the value indicates that data is from no more than two systems. This implies that the DCUSYSID variable (which is specified above as the value of the IDVAR= parameter) should contain no more than two values.

RANGES=*number-of-ranges*
represents the maximum number of ranges that can occur during this execution of %CPPROCES or %CMPROCES. A new range is created when the difference between the datetime stamps of two consecutive records exceeds the value of the INT= parameter. This break is referred to as a gap in the data. In this example, the specified value is a maximum of three. This means there can be no more than two gaps greater than 25 hours (the value of the INT= parameter).

KEEP=*number-of-weeks*
represents the maximum number of weeks for which you want to retain control data. A range is aged out (removed) when the end-of-range datetime is older than allowed by the value of this parameter. In this example, the specified number of weeks is nine.

FORCE=NO | YES
FORCE=NO indicates that duplicate data should not be processed into the PDB.

TERM=NO | YES
TERM=NO indicates that processing should not stop if duplicate data is encountered.

For a complete description of these and other parameters and appropriate values for each collector, see "%CPDUPCHK" on page 86.

Review and modify your call as follows:

- For the value of the SOURCE= parameter, specify the three-character identifier that you chose earlier.
- For the value to use for the ENDFILE= parameter, look in the MXG staging code for your data collector.
- For the value to use for the IDVAR= parameter, look in the MXG staging code for your data collector.
- For the value to use for the TIMESTMP= parameter, look in the MXG staging code for your data collector.
- Review the values of the INT=, SYSTEMS=, RANGES=, and KEEP= parameters, and change those values to be appropriate for your site.
- Review the TERM= and FORCE= parameters, and change those values to be appropriate for your purpose. (Note that the combination TERM=YES,FORCE=YES is not meaningful.) The values should correspond to the

DUPMODE= parameter that you will use on the call to %CMPROCES or %CPPROCES.

Insert Calls and Include Member or File

The instructions (to insert the call to %CPDUPINT and the call to %CPDUPDSN, and to include the member or file that contains the call to %CPDUPCHK) depend on the version of MXG that you are running.

- If you are running a version of MXG prior to version 16.04, use Instructions Set #1.
- If you are running MXG version 16.04 or later,
  - use Instructions Set #1 if you are using the IMAC member to specify your data source customizations.
  - use Instructions Set #2 if you are using the IMACKEEP member to specify your data source customizations.

Instructions Set #1

1 Insert a call to %CPDUPINT.

a In MXG.MXG.SOURCLIB, in member or file IMACAAAA, there is a table with the names of the other IMAC members or files and a description of what is in each.

Find the name of the IMAC member or file for your data collector. For example, the member IMACDCOL is for DCOLLECT data.

b If your data collector's IMAC member or file is not already in MXG.USERID.SOURCLIB, copy that member or file from MXG.MXG.SOURCLIB to MXG.USERID.SOURCLIB. Keep the same name.

c Insert a call to %CPDUPINT: At the beginning of your data collector's IMAC member or file in MXG.USERID.SOURCLIB, insert a call to the %CPDUPINT macro. Use the following form:

```
%CPDUPINT ;
```

Note that the %CPDUPINT macro does not have any parameters.

2 Insert a call to %CPDUPDSN.

a In that same IMAC member or file, inside any one (but only one) of the _L macro definitions, insert a call to the %CPDUPDSN macro. Use the form

```
%%CPDUPDSN(SOURCE=xxx)
```

where *xxx* is a three-character identifier of your data source or data collector. You can choose any identifier as long as it is not one of the identifiers that is on the list of values for the SOURCE= parameter in the %CPDUPCHK macro. Note that the call has an unusual form: there are two leading percent signs and no trailing semicolon.

For example, in IMACDCOL, you might change

```
MACRO _LDCODSN DCOLDSET %
MACRO _KDCODSN          %
MACRO _LDCOCLU DCOLCLUS %
...
```

to

```
MACRO _LDCODSN %%cpdupdsn(source=DCO)
              DCOLDSET %
MACRO _KDCODSN          %
MACRO _LDCOCLU DCOLCLUS %
...
```

3 Include the member or file that contains the call to %CPDUPCHK.

a In MXG.MXG.SOURCLIB, in member or file IMACAAAA, there is a list of the names of the members or files and a description of what is in each.

Find the name of the member or file that decodes data for your data collector. For example, for DCOLLECT data the member or file is named VMACDCOL.

b In MXG.MXG.SOURCLIB, scan through your data collector's VMAC member or file for a reference to an IHDR member or file.

For example, for DCOLLECT data in MXG version 14.02 and later, VMACDCOL refers to a member or file that is called IHDRDCOL.

c If the VMAC member or file *does* refer to an IHDR member or file, include the member or file that contains the call to %CPDUPCHK in the IHDR member or file.

i If the member or file with the IHDR name is not already in MXG.USERID.SOURCLIB, copy that member or file from MXG.MXG.SOURCLIB to MXG.USERID.SOURCLIB. Keep the same name.

ii At the beginning of that member or file in MXG.USERID.SOURCLIB (typically, the member or file is empty), insert the following line of code:

- On z/OS,

```
%INCLUDE
'fully-qualified-PDS($$GENxxx)' ;
```

- On UNIX,

```
%INCLUDE
'full-path-directory/gen_xxx.sas' ;
```

Remember that directory names and filenames are case sensitive on UNIX.

- On Windows,

```
%INCLUDE
'full-path-directory\gen_xxx.sas' ;
```

For *fully-qualified-PDS* or *full-path-directory*, substitute the location of MXG.USERID.SOURCLIB. The member or file name refers to the member or file that you created above. The *xxx* refers to the three-character identifier that you chose above.

Note: The %INCLUDEs above use absolute locations. If you prefer to use relative locations, use the following code.

- On z/OS:

```
%INCLUDE SOURCLIB ($$GENxxx)
;
```

- On UNIX:

```
%INCLUDE SOURCLIB (gen_xxx) ;
```

- On Windows:

```
%INCLUDE SOURCLIB (gen_xxx) ;
```

The SOURCLIB variable is a "concatenation" of the values in the MXGSRC= parameter on %CPSTART. Note that you must *not* add the .sas extension on UNIX or Windows. The extension will be appended automatically on UNIX and Windows. △

d If the VMAC member or file *does not* refer to an IHDR member or file, include the member or file that contains the call to %CPDUPCHK in every one of the data collector's EX members or files.

i Identify all of the EX member or files for your data collector. To find the member names, scan the data collector's VMAC member or file in MXG.MXG.SOURCLIB and look for %INCLUDE statements that refer to members of SOURCLIB that have the prefix EX.

ii For each EX member or file, if a member or file with the same name does not already exist in MXG.USERID.SOURCLIB, copy the EX member or file from MXG.MXG.SOURCLIB into MXG.USERID.SOURCLIB. Keep the same name.

iii In each of these EX members or files in MXG.USERID.SOURCLIB, there is an OUTPUT statement. In front of the OUTPUT statement, on a new line insert this line of code.

□ On z/OS:

```
%INCLUDE 'fully-qualified-PDS($$GENxxx)' ;
```

□ On UNIX:

```
%INCLUDE 'full-path-directory/gen_xxx.sas' ;
```

Remember that directory names and filenames are case sensitive on UNIX.

□ On Windows:

```
%INCLUDE 'full-path-directory\gen_xxx.sas'
;
```

For *fully-qualified-PDS* and *full-path-directory*, substitute the location of MXG.USERID.SOURCLIB. The member name or filename refers to the member or file that you created above. The *xxx* refers to the three-character identifier that you chose above.

Note: The above %INCLUDEs use absolute locations. If you prefer to use relative locations, use the following code.

□ On z/OS:

```
%INCLUDE SOURCLIB ($$GENxxx)
;
```

□ On UNIX:

```
%INCLUDE SOURCLIB (gen_xxx) ;
```

□ On Windows:

```
%INCLUDE SOURCLIB (gen_xxx) ;
```

The SOURCLIB variable is a "concatenation" of the values in the MXGSRC= parameter on %CPSTART. Note that you must *not* add the .sas extension on UNIX or Windows. The extension will be appended automatically on UNIX and Windows. △

Note: Most collectors have several OUTPUT statements (one OUTPUT statement per EX member) and thus the potential for calling %CPDUPCHK several times. The duplicate-data-checking macros have verification routines to ensure that %CPDUPCHK executes only once and checks the data only once. △

Instructions #2

1 Insert a call to the %CPDUPINT macro.

   a If the IMACKEEP member or file is not already in MXG.USERID.SOURCLIB, copy it from MXG.MXG.SOURCLIB to MXG.USERID.SOURCLIB. Keep the same name.
   b In the IMACKEEP member or file in MXG.USERID.SOURCLIB, insert a call to the %CPDUPINT macro immediately after the %INCLUDE statement for IMACOLDV. Use the form

```
%CPDUPINT ;
```

   Note that the %CPDUPINT macro does not have any parameters.

2 Insert a call to the %CPDUPDSN macro.

   a Also in the IMACKEEP member or file in MXG.USERID.SOURCLIB, there is a group of _W macro definitions for your data collector.
   b Copy any one (but only one) of these _W macro definitions and insert it immediately after the call to the %CPDUPINT macro.
   c Inside that copy, insert a call to the %CPDUPDSN macro. Use the form

```
%%CPDUPDSN(SOURCE=xxx)
```

   where *xxx* is the three-character identifier that you chose above. Note that the call has an unusual form: there are two leading percent signs and no trailing semicolon.

   For example, for DCOLLECT you might copy the macro definition

```
MACRO _WDCODSN DCOLDSET%
```

   and insert the call so that the copy looks like this:

```
MACRO _WDCODSN %%CPDUPDSN(SOURCE=DCO)
              DCOLDSET %
```

3 Include the member or file that contains the call to the %CPDUPCHK macro.

   a In MXG.MXG.SOURCLIB, in member or file IMACAAAA, there is a list of the names of the members or files and a description of what is in each.

   Find the name of the member or file that decodes data for your data collector. For example, for DCOLLECT data the member or file is named VMACDCOL.
   b Scan through your data collector's VMAC member or file for a reference to an IHDR member or file.

   For example, for DCOLLECT data in MXG version 14.02 and later, VMACDCOL refers to a member or file that is called IHDRDCOL.
   c If the VMAC member or file *does* refer to an IHDR member or file, include the member or file that contains the %CPDUPCHK call into the IHDR member or file.

      i If the member or file with the IHDR name is not already in MXG.USERID.SOURCLIB, copy that member or file from MXG.MXG.SOURCLIB to MXG.USERID.SOURCLIB. Keep the same name.

ii At the beginning of that member or file in MXG.USERID.SOURCLIB (typically, the member or file is empty), insert the following line of code.

- On z/OS:

```
%INCLUDE
'fully-qualified-PDS($$GENxxx)' ;
```

- On UNIX:

```
%INCLUDE
'full-path-directory/gen_xxx.sas' ;
```

  Remember that directory names and filenames are case sensitive on UNIX.

- On Windows:

```
%INCLUDE 'full-path-directory\gen_xxx.sas'
;
```

For *fully-qualified-PDS* or *full-path-directory*, substitute the location of MXG.USERID.SOURCLIB. The member name or filename refers to the member or file that you created above. The *xxx* refers to the three-character identifier that you chose above.

Note: The above %INCLUDEs use absolute locations. If you prefer to use relative locations, use the following code.

- On z/OS:

```
%INCLUDE SOURCLIB ($$GENxxx)
;
```

- On UNIX:

```
%INCLUDE SOURCLIB (gen_xxx) ;
```

- On Windows:

```
%INCLUDE SOURCLIB (gen_xxx) ;
```

The SOURCLIB variable is a "concatenation" of the values in the MXGSRC= parameter on %CPSTART. Note that you must *not* add the .sas extension on UNIX or Windows. The extension will be appended automatically on UNIX and Windows. △

d If the VMAC member or file *does not* refer to an IHDR member or file, include the member or file that contains the %CPDUPCHK call into _E macro definitions in the IMACKEEP member or file.

i In your data collector's VMAC member or file in MXG.USERID.SOURCLIB, there is a group of _E macro definitions.

ii Copy all of these _E macro definitions from the VMAC member or file to the beginning of the IMACKEEP member or file.

iii Inside every one of these _E macro definitions in the IMACKEEP member or file, include the member or file that contains the call to %CPDUPCHK. Use the following form.

- On z/OS:

```
%%INCLUDE 'fully-qualified-PDS($$GENxxx)' ;
```

- On UNIX:

```
%%INCLUDE 'full-path-directory/gen_xxx.sas' ;
```

Remember that directory names and filenames are case sensitive on UNIX.

- On Windows:

```
%%INCLUDE
'full-path-directory\gen_xxx.sas' ;
```

For *fully-qualified-PDS* and *full-path-directory*, substitute the location of MXG.USERID.SOURCLIB. The member name or filename refers to the member or file that you created above. The *xxx* refers to the three-character identifier that you chose above.

Note: The above %INCLUDEs use absolute locations. If you prefer to use relative locations, use the following code.

- On z/OS:

```
%%INCLUDE SOURCLIB ($$GENxxx)
;
```

- On UNIX:

```
%%INCLUDE SOURCLIB (gen_xxx) ;
```

- On Windows:

```
%%INCLUDE SOURCLIB (gen_xxx) ;
```

The SOURCLIB variable is a "concatenation" of the values in the MXGSRC= parameter on %CPSTART. Note that you must *not* add the .sas extension on UNIX or Windows. The extension will be appended automatically on UNIX and Windows. △

For example, change

```
MACRO _EDCODSN
      %%INCLUDE SOURCLIB(EXDCODSN);
      %
```

to

```
MACRO _EDCODSN
      %%INCLUDE SOURCLIB($$GENDCO);
      %%INCLUDE SOURCLIB(EXDCODSN);
      %
```

Note: Although most collectors have several _E macros and, therefore, several %CPDUPCHK calls, the duplicate-data-checking macros have verification routines to ensure that the macro executes only once and checks the data only once. △

User-Generated Table Definitions and User-Generated Non-MXG-Based Staging Code

Use these instructions if you

- process data by calling %CMPROCES or %CPPROCES
- on the call to %CMPROCES or %CPPROCES, set the COLLECTR= parameter to GENERIC or to a collector name that is used as a tag (along with tool name) on

collector-support entities that will be or have been packaged and installed (by using %CPPKGCOL and %CPINSPKG) for use with %CPPROCES

- on the call to %CMPROCES or %CPPROCES, use user-generated code to do the collector-specific staging of the data that will be processed into the user-generated tables.

For information on using the Generic Collector Facility with non-MXG-based staging code, see the chapter "Setup: Generic Collector Facility" in the *SAS IT Resource Management User's Guide*.

Here is an overview of the preparation for duplicate-data checking in this case:

1. On your call to the %CMPROCES or %CPPROCES macro, set the DUPMODE= parameter to FORCE, DISCARD, or TERMINATE.
2. Create a member or file for a call to the %CPDUPCHK macro.
3. In that member or file, add the %CPDUPCHK call, and modify the call as necessary.
4. Insert a call to the %CPDUPINT macro in your collector-specific staging code.
5. Insert a call to the %CPDUPDSN macro in your collector-specific staging code.
6. Include into your collector-specific staging code the member or file that contains the call to the %CPDUPCHK macro.
7. In the non-collector-specific staging code, the %CMPROCES or %CPPROCES macro will automatically call the %CPDUPUPD macro.

The following sections describe, in detail, the steps that you perform.

Set the DUPMODE= Parameter

On the %CMPROCES or %CPPROCES macro call, set the DUPMODE= parameter to TERMINATE, DISCARD, or FORCE. Each of these values indicates that you do want to use duplicate-data checking. The particular value that you specify indicates how you intend to set the FORCE= and TERM= parameters on the call to the %CPDUPCHK macro.

- To indicate that you intend to stop if duplicate data is encountered, set DUPMODE=TERMINATE.
- To indicate that you intend to cause processing to continue but duplicate data to be rejected if duplicate data is encountered, specify DUPMODE=DISCARD.
- To indicate that you intend to cause processing to continue and duplicate data to be accepted if duplicate data is encountered, specify DUPMODE=FORCE. (Use this setting only when loading data that is not duplicate, even though it may be in the same datetime range and for the same machine or system as previously loaded data.)

Note: Do not specify DUPMODE=INACTIVE. (Also, do not omit specifying the DUPMODE= parameter, because DUPMODE= defaults to INACTIVE.)

If DUPMODE= is INACTIVE, your calls to the %CPDUPINT, %CPDUPDSN, and %CPDUPCHK macros will run, but %CMPROCES or %CPPROCES will not call the %CPDUPUPD macro. △

Create the Member or File

If you have MXG, it has a convenient storage place for the member or file. Otherwise, use a storage location that you name.

- *If you have MXG:* On the host where you will process the data, if there is no member or file in MXG.USERID.SOURCLIB with the following name, create one. Keep the same name.
  - On z/OS, member **$$GENxxx**.
  - On UNIX, file **gen_xxx.sas**.
  - On Windows, file **gen_xxx.sas**.
- *If you do not have MXG:* On the host where you will process the data,
  - On z/OS, create a PDS and, in it, create a member named **$$GENxxx**.
  - On UNIX, create a directory and, in it, create a file named **gen_xxx.sas**.
  - On Windows, create a directory and, in it, create a file named **gen_xxx.sas**.

The xxx is a three-character identifier for your data source or data collector. You can choose any identifier as long as it is not one of the identifiers on the list of values for the SOURCE= parameter in "%CPDUPCHK" on page 86.

Modify the Member or File

In this member or file, if there is no call to the %CPDUPCHK macro, add a %CPDUPCHK call that is similar to the following code:

```
%CPDUPCHK(SOURCE=ABC,IDVAR=SYSTEMID,TIMESTMP=DTIME,
     ENDFILE=LAST,INT=00:10,SYSTEMS=4,RANGES=4,KEEP=53);
```

(If you are running on z/OS, you will find some additional %CPDUPCHK examples in the SAS IT Resource Management PDS that is named CPMISC.)

The parameters and values that are used in the this example are defined below:

SOURCE=*identifier*
specifies a three-character identifier for your data source or data collector. This example uses the value ABC.

IDVAR=*variable-name*
specifies the name of the SAS variable that identifies the system or machine from which the data originated. This example uses the value SYSTEMID.

TIMESTMP=*timestamp-variable-name*
specifies the name of the SAS variable that contains the timestamp for each input record. In this example, the value is DTIME.

ENDFILE=*variable-name*
identifies the SAS variable that represents the end-of-file condition for your INFILE statement. In this example, the value is LAST.

INT=*interval*
represents the maximum interval that is allowed between the datetime stamps on any two consecutive input records from the same system or machine. If the interval between the datetime stamp values exceeds the value of this parameter, then a new range is created (see the RANGES= parameter). In this example, a range can contain gaps no larger than 10 minutes.

SYSTEMS=*number-of-systems*
represents an estimate of the maximum number of systems or machines that collected data for your input file. In this example, the value indicates that data is from no more than four systems. This implies that the SYSTEMID variable (which is specified above as the value of the IDVAR= parameter) should contain no more than four values.

RANGES=*number-of-ranges*
represents the maximum number of ranges that can occur while the data is being read. A new range is created when the difference between the datetime stamps of two consecutive records exceeds the value of the INT= parameter. This break is referred to as a gap in the data. In this example the specified value is four. This means that there can be no more than three gaps greater than 10 minutes (the value of the INT= parameter).

KEEP=*number-of-weeks*
represents the maximum number of weeks for which you want to retain control data. A range is aged out (removed) when the end-of-range datetime is older than permitted by the value of this parameter. In this example, the specified value is 53 weeks.

For a complete description of these and other parameters for this macro, see "%CPDUPCHK" on page 86.

Review and modify your call as follows:

- For the value of the SOURCE= parameter, specify the three-character identifier that you chose earlier.
- For the value of the ENDFILE= parameter, look in your collector-specific staging code.
- For the value to use for the IDVAR= parameter, look in your collector-specific staging code.
- For the value to use for the TIMESTMP= parameter, look in your collector-specific staging code.
- Review the values of the INT=, SYSTEMS=, RANGES=, and KEEP= parameters, and change those values to be appropriate for your site.
- Review the TERM= and FORCE= parameters, and change those values to be appropriate for your purpose. (Note that the combination TERM=YES,FORCE=YES is not meaningful.) The values should correspond to the DUPMODE= parameter that you will use on the call to %CMPROCES or %CPPROCES.

Insert a Call to %CPDUPINT

In your staging code, insert a call to the %CPDUPINT macro in front of the DATA statement that initiates the DATA step that reads your raw data. Use the following form:

```
%CPDUPINT ;
```

Note that the %CPDUPINT macro does not have any parameters.

Insert a Call to %CPDUPDSN

For the call, use the form

```
%CPDUPDSN(SOURCE=xxx)
```

where *xxx* is a the three-character source name that you chose above for this collector. Note that the call has an unusual form: there is no trailing semicolon.

The call to the %CPDUPDSN macro must be inserted between the word **DATA** and the character **;** in the DATA statement that initiates the DATA step that reads the raw data and writes the staged data. For example, suppose you have a DATA step that reads raw data, and writes the staged data to GENLIB.STAGED:

```
DATA GENLIB.STAGED ;
   ...
   OUTPUT GENLIB.STAGED ;
```

You would change that to

```
DATA GENLIB.STAGED %cpdupdsn(source=xxx) ;
   ...
   OUTPUT GENLIB.STAGED ;
```

Note: Do not use the OUTPUT statement in this form:

```
OUTPUT ;
```

Any OUTPUT statement to the data set or view to which the data is being staged (in this case, GENLIB.STAGED) must explicitly specify the data set or view. Use the OUTPUT statement in this form:

```
OUTPUT libref.data-set-or-view-name;
```

△

Include the Member or File That Contains the Call to %CPDUPCHK

In your staging code, place the %INCLUDE statement after the TIMESTMP and IDVAR variables have been read in from your raw data and before any logic that would execute an OUTPUT statement. Within that range, place the %INCLUDE statement as early in your code as possible in order to avoid any unnecessary processing of data that is discovered to be duplicate data.

Point the %INCLUDE statement to the member or file that you created above. Use the following form.

- On z/OS, insert the following line in your staging code:

  ```
  %INCLUDE 'fully-qualified-PDS($$GENxxx)' ;
  ```

- On UNIX, insert the following line in your staging code:

  ```
  %INCLUDE 'full-path-directory/gen_xxx.sas' ;
  ```

  Remember that directory names and filenames are case sensitive on UNIX.

- On Windows, insert the following line in your staging code:

  ```
  %INCLUDE 'full-path-directory\gen_xxx.sas' ;
  ```

For *fully-qualified-PDS* or *full-path-directory*, substitute the location of MXG.USERID.SOURCLIB. The *xxx* refers to the three-character identifier that you chose above.

Note: The above %INCLUDEs use absolute locations. Use the following instructions if you prefer to use relative locations.

- If you stored the %CPDUPCHK call in MXG.USERID.SOURCLIB, use the following code:
  - On z/OS:

    ```
    %INCLUDE SOURCLIB ($$GENxxx)
    ```

  - On UNIX:

    ```
    %INCLUDE SOURCLIB (gen_xxx)
    ```

  - On Windows:

```
%INCLUDE SOURCLIB
(gen_xxx)
```

The SOURCLIB fileref is a "concatenation" of the values in the MXGSRC= parameter on %CPSTART. Note that you must *not* add the .sas extension on UNIX or Windows. The extension will be appended automatically on UNIX and Windows.

- If you did not store the %CPDUPCHK call in MXG.USERID.SOURCLIB, create a fileref for your storage location by using the SAS FILENAME statement

```
FILENAME myfref 'fully-qualified-PDS-name-or-full-path-directory-name';
```

where *myfref* is the name that you give to this fileref. After you create the fileref, use the following code:

- On z/OS:

```
%INCLUDE myfref
($$GENxxx)
```

- On UNIX:

```
%INCLUDE myfref (gen_xxx)
```

- On Windows:

```
%INCLUDE myfref (gen_xxx)
```

Note that you must *not* add the .sas extension on UNIX or Windows. The extension will be appended automatically on UNIX and Windows.

△

APPENDIX

5

Variable Interpretation Types

Variable Interpretation Types

An interpretation type (for example, INT, C2RATE, STRING, DATE, and so on) is assigned to a regular or derived variable, and this assignment is based on the characteristics of the data in that variable. (Formula variables have an interpretation type of FORMULA.) The interpretation type describes the characteristics of the data in the variable and determines the default settings for a number of characteristics of the variable, such as default length, format, and statistics.

In tables that are supplied by SAS IT Resource Management, the interpretation type is already assigned to each of the variables. However, if you add a variable to a supplied table or create a new table of variables, then you must assign interpretation types to your new variables.

Assigning the appropriate interpretation type to a variable is necessary not only for proper formatting of reports but also for proper processing and reduction. A variable's interpretation type provides or implies the following (for a complete summary, see "Variable Interpretation Type Summary Table" on page 710):

- which default statistics to keep for the variable and how to calculate them. This is probably the most important implication of interpretation type.
- the default data type of the variable (character or numeric).
- the default maximum length of the variable.
- which default SAS format to use when displaying the data.

SAS IT Resource Management supports two types of tables: INTERVAL and EVENT. The statistics that are associated or calculated for an interpretation type depend on whether the variable is contained in an interval table or an event table.

An INTERVAL table records information that is collected over a (usually fixed) time interval. Thus, interval tables must have a DURATION variable. For example, an INTERVAL table that records paging subsystem data might include data such as the number of page faults that occur over the life of the interval (duration) and the amount of paging space that is used over the life of the interval.

An EVENT table records information that is collected for an asynchronous event. For example, an EVENT table that records task termination events might include data such as the time of the task termination, the number of page faults over the life of the task, and the number of I/O operations that are performed by the task.

The datetime stamp that is included in an event table records when the event occurs. The datetime stamp that is included in an interval table record usually represents the start of the interval.

Interpretation types are described below, along with their default information, such as type, length, format, and interval and event statistics. The definitions for the statistics codes are provided immediately after the interpretation type definitions. You can also refer to the summary of variable interpretation types in the "Variable Interpretation Type Summary Table" on page 710. This table enables you to easily browse through the interpretation types with their default statistics and format. Note that the maximum length of numeric variables is eight and the maximum length of character variables is 200.

AVERAGE Arithmetic mean

Default Type: Numeric

Default Length: 8

Default Format: BEST12.2

Default Interval Statistics: Aw

Default Event Statistics: csA

C2RATE Counter to rate. The interpretation type C2RATE is used by the process step in SAS IT Resource Management. This interpretation type converts a COUNTER to a RATE in an INTERVAL table. The conversion takes place in the non-collector-specific staging code.

For example, if you have recorded odometer mileage and the start time of each mileage reading, then the counter is mileage. During the process step, the duration is determined by calculating the difference between the start times of one observation and the preceding observation. Then, the number of miles per second can be calculated for each observation. To calculate the number of miles per second, find the difference between the odometer mileage of one observation and the mileage of the preceding observation, and then divide the result by the duration in seconds.

Default Type: Numeric

Default Length: 8

Default Format: BEST12.2

Default Interval Statistics: Aw

Default Event Statistics: -

A counter continues increasing across all intervals until it reaches its maximum value and then, typically, it resets to zero and starts again. The counter maximums are 65,536 (for a 16-bit counter), 4,294,967,296 (for a 32-bit counter), and 18,446,744,073,709,551,616 (for a 64-bit counter). Using the previous and current value of the counter and knowing the maximum limit for that particular counter, it is possible to calculate the rate for that interval, based on these assumptions: if the previous value is smaller than the current value, it is assumed that the counter has not reset; if the previous value is larger than the current value, it is assumed that the counter has reset one time.

However, a problem that has to be dealt with, especially for 16-bit counters, is that fast-moving counters can reset more than one time during an interval. One way to resolve this problem is to sample at

a shorter interval so that there is not time during the interval for the counters to reset more than one time.

SAS IT Resource Management also provides another way to resolve this problem. Although SAS IT Resource Management converts counters to rates automatically, there are three macro variables that you can use to affect how the conversion is done. By default, these macro variables are set as follows:

- Macro variable CP16PCT is for use with 16-bit counters. By default, it is set as follows:

  ```
  %let cp16pct = 0.95 ;
  ```

  As a result of this setting, a threshold is calculated for 16-bit counters, and the threshold has the value of 0.95 * 65536 = 62259.2.
- Macro variable CP32PCT is for use with 32-bit counters. By default, it is set as follows:

  ```
  %let cp32pct = 0.97 ;
  ```

  As a result of this setting, a threshold is calculated for 32-bit counters, and the threshold has the value of 0.97*4292967296 = 4164178277.
- Macro variable CP64PCT is for use with 64-bit counters. By default, it is set as follows:

  ```
  %let cp64pct = 0.97 ;
  ```

  As a result of this setting, a threshold is calculated for 64-bit counters, and the threshold has the value of 0.97*18446744073709551616 = 17893341751498265067.

When a rate is being determined and the current value is less than the previous value, then

- if the previous value is greater than or equal to the threshold, the rate is calculated based on the assumption that one reset occurred during the interval. That is, the previous value was so close to the counter's maximum that it is understandable it might reset (one time) during the interval.

  In this case, a "Corrected" message (see example below) is written to the SAS log.
- if the previous value is less than the threshold, the rate is not calculated and the rate is set, instead, to missing. That is, the previous value is so far from the counter's maximum that the counter must have been moving extremely fast to reset during the interval. And if the counter is moving that fast, it is at risk of having reset more than one time during that interval.

  In this case, a "Set to missing" message (see example below) is written to the SAS log.

The following example illustrates the situation for a 16-bit counter:

```
Counter value   Calculated Rate (assuming 5 min intervals)
    20000                .
    40000            66.67
```

```
62260           72.20
  300           11.92
40000          132.33
62259           74.19
 2000             .
```

The first observation's rate is missing, because there is no previous value with which to calculate the difference.

The first time the counter resets, the value preceding the reset (62260) meets (or exceeds) the default threshold of 95% for 16-bit counters (62259.2). The rate is therefore calculated (based on one reset) and the following message is written to the SAS log:

```
(CLEANUP)Obs4 16 bit Overflow  Start 01JAN01:00:10:00 SVAL=62260
     Corrected        End 01JAN2001:00:15:00.00 SVAL=300
```

The second time the counter resets, the value preceding the reset (62259) does not meet (or exceed) the default threshold of 95% for 16-bit counters (62259.2). The rate is therefore set to missing and the following message is written to the SAS log:

```
(CLEANUP)Obs 7 Inconsistent  Start 01JAN01:00:25:00 SVAL=62259
     Set to missing   End 01JAN2001:00:30:00.00 SVAL=2000
```

If a large percentage of your data is producing the second type of message, the work measured by the counters may have increased so much that the counters are resetting more than one time within an interval. Take one or both of the following actions:

- reduce the interval at which you sample, although this will increase the volume of data that you collect
- lower the thresholds. For example, you could insert the following SAS statement in the process-and-reduce job before the call to the process macro:

  ```
  %let cp16pct=0.94;
  ```

  Note: As you lower the thresholds, more rates are calculated, but the rates may be incorrect because of multiple resets due to fast-moving counters. △

  Note: Refer to the SAS Companion documentation for your host for information on "Length and Precision of Variables" if you use any 64-bit counters. △

COUNT

Count of events. The COUNT interpretation is an accumulated count of the number of events, starting at zero, for the interval or event. An everyday example of a COUNT is the number of page faults that occur in a job.

Default Type: Numeric

Default Length: 8

Default Format: BEST12.

Default Interval Statistics: SA

Default Event Statistics: cSA

When an average is calculated on a COUNT in an interval table, the resulting value is always a RATE. This is because the only thing that can be used to normalize the interval observations is the DURATION. Hence, COUNT/DURATION= RATE, because there is

no way to guarantee that the duration of all intervals will always be the same.

COUNTER — Incremental counter across interval boundaries. A COUNTER is an accumulated count of events that is not reset to zero (that is, it spans events or intervals). An everyday example of a COUNTER is an odometer, which resets to zero only when the available digits are used up. Interpretation type COUNTER has very limited use.

Default Type: Numeric
Default Length: 8
Default Format: BEST12.
Default Interval Statistics: -
Default Event Statistics: -

DATE — SAS date value (number of days since 1Jan60)

Default Type: Numeric
Default Length: 8
Default Format: DATE9.
Default Interval Statistics: -
Default Event Statistics: -

DATETIME — SAS datetime value (number of seconds since midnight 1Jan60)

Default Type: Numeric
Default Length: 8
Default Format: DATETIME21.2
Default Interval Statistics: -
Default Event Statistics: -

DECADDRESS — DECNet network device address

Default Type: Character
Default Length: 200
Default Format: .
Default Interval Statistics: -
Default Event Statistics: -

D2RATE — DELTA to rate. The interpretation type D2RATE is used to convert a DELTA (a count of something) to a RATE in an INTERVAL table. The conversion takes place in the non-collector-specific staging code. A DELTA is a variable that represents the difference between two monotonically increasing or decreasing values.

For example, a baseball scoreboard might display the inning, the starting time of the inning, and the runs that are scored in each inning. The “runs” is a DELTA. The duration would be calculated by finding the difference between the starting times of one observation and the preceding observation. Then, the runs per second would be calculated for each observation, by dividing the number of runs in an interval by the duration in seconds.

Default Type: Numeric
Default Length: 8
Default Format: BEST12.2.
Default Interval Statistics: Aw

Default Event Statistics: -

ENUM — Enumeration number (number of items within a set)

Default Type: Numeric
Default Length: 4
Default Format: 5.
Default Interval Statistics: -
Default Event Statistics: -

FLOAT — Floating-point number

Default Type: Numeric
Default Length: 8
Default Format: BEST12.2
Default Interval Statistics: -
Default Event Statistics: -

FORMULA — A formula interpretation applies the specified formula source statements to the observation.

This interpretation type cannot be selected directly. It is a consequence of defining a formula variable.

Default Type: N/A
Default Length: 8
Default Format: N/A
Default Interval Statistics: N/A
Default Event Statistics: N/A

GAUGE — A GAUGE is a number that represents an instantaneous reading. An everyday example of a GAUGE is the reading on a car's speedometer.

Default Type: Numeric
Default Length: 6
Default Format: BEST12.2
Default Interval Statistics: csA
Default Event Statistics: csA

GAUGE differs from INT; statistics are gathered for a GAUGE and not for an INT. GAUGE differs from COUNT in how the average is calculated. In the case of a GAUGE, the average is a simple average (not divided by duration), because it is an instantaneous reading and does not represent an entire interval.

HEXFLAGS — Binary data flags

Default Type: Character
Default Length: 200
Default Format: $HEX.
Default Interval Statistics: -
Default Event Statistics: -

INT — Integer number. An INT is a number that stands by itself and for which it makes no sense to perform statistical calculations. Examples of an INT are a hardware model number, the number of CPUs installed, and the version/release number of a piece of software.

Default Type: Numeric
Default Length: 6
Default Format: BEST12.
Default Interval Statistics: -
Default Event Statistics: -

IPADDRESS Network device address
Default Type: Character
Default Length: 15
Default Format: .
Default Interval Statistics: -
Default Event Statistics: -

LABEL SAS label for a table or variable
Default Type: Character
Default Length: 40
Default Format: .
Default Interval Statistics: -
Default Event Statistics: -

MAXIMUM Numeric maximum
Default Type: Numeric
Default Length: 8
Default Format: BEST12.2
Default Interval Statistics: X
Default Event Statistics: X

MINIMUM Numeric minimum
Default Type: Numeric
Default Length: 8
Default Format: BEST12.2
Default Interval Statistics: N
Default Event Statistics: N

NAME String that contains one word
Default Type: Character
Default Length: 64
Default Format: .
Default Interval Statistics: -
Default Event Statistics: -

NETADDRESS TCP/IP network device address
Default Type: Character
Default Length: 11
Default Format: .
Default Interval Statistics: -
Default Event Statistics: -

PCTGAUGE In event-type data, a number with values that are between 0 and 1. In interval-type data, a number with values that are between 0 and 1 and that do not need to be weighted.

More specifically, its value is a "snapshot" taken at the time of an event or at some one time during an interval. In the raw data (or at least by the time the raw data has been staged by the collector-specific staging code), the value is equal to or greater than zero and equal to or less than one.

Default Type: Numeric

Default Length: 6

Default Format: PERCENT9.2

Default Interval Statistics: csA

Default Event Statistics: csA

For example, a value of 1 with an interpretation type of PCTGAUGE is stored as 1 and is displayed as 100.00%.

Choose between PCTGAUGE and PCTGAUGE100 based on the range of values. If the range is between 0 and 1, use PCTGAUGE. If the range is between 1 and 100, use PCTGAUGE100.

Choose between PCTGAUGE and PERCENT based on weighting. Event-type data is not weighted, so there is no difference. Interval-type data is not weighted if you use PCTGAUGE and is weighted if you use PERCENT.

PCTGAUGE100 In event-type data, a number with values that are between 0 and 100. In interval-type data, a number with values that are between 0 and 100 and that do not need to be weighted.

More specifically, its value is a "snapshot" taken at the time of an event or at some one time during an interval. In the raw data (or at least by the time the raw data has been staged by the collector-specific staging code), the value is equal to or greater than zero and equal to or less than one hundred.

Default Type: Numeric

Default Length: 6

Default Format: 6.2

Default Interval Statistics: csA

Default Event Statistics: csA

For example, a value of 1 with an interpretation type of PCTGAUGE100 is stored in an event-type table as 1 and is displayed as 1.00.

Choose between PCTGAUGE100 and PCTGAUGE based on the range of values. If the range is between 0 and 1, use PCTGAUGE. If the range is between 1 and 100, use PCTGAUGE100.

Choose between PCTGAUGE100 and PERCENT100 based on weighting. Event-type data is not weighted, so there is no difference. Interval-type data is not weighted if you use PCTGAUGE100 and is weighted if you use PERCENT100.

PERCENT In event-type data, a number with values that are between 0 and 1. In interval-type data, a number with values that are between 0 and 1 and that do need to be weighted.

More specifically, its value is not a "snapshot." In the raw data (or at least by the time the raw data has been staged by the

collector-specific staging code), the value is equal to or greater than zero and equal to or less than one. If the table is an interval-type table, the reduction step weights the calculation by the alternate weight variable (or by DURATION by default).

Default Type: Numeric

Default Length: 6

Default Format: PERCENT9.2

Default Interval Statistics: Aw

Default Event Statistics: csA

For example, a value of 1 with an interpretation type of PERCENT is stored in an event-type table as 1 and is displayed as 100.00%.

Choose between PERCENT and PERCENT100 based on the range of values. If the range is between 0 and 1, use PERCENT. If the range is between 1 and 100, use PERCENT100.

Choose between PERCENT and PCTGAUGE based on weighting. Event-type data is not weighted, so there is no difference. Interval-type data is not weighted if you use PCTGAUGE and is weighted if you use PERCENT.

PERCENT100 In event-type data, a number with values that are between 0 and 100. In interval-type data, a number with values that are between 0 and 100 and that do need to be weighted.

More specifically, its value is not a "snapshot." In the raw data (or at least by the time the raw data has been staged by the collector-specific staging code), the value is equal to or greater than zero and equal to or less than one hundred. If the table is an interval-type table, the reduction step weights the calculation by the alternate weight variable (or by DURATION by default).

Default Type: Numeric

Default Length: 6

Default Format: 6.2

Default Interval Statistics: Aw

Default Event Statistics: csA

For example, a value of 1 with an interpretation type of PERCENT100 is stored in an event-type table as 1 and is displayed as 1.00.

Choose between PERCENT100 and PERCENT based on the range of values. If the range is between 0 and 1, use PERCENT. If the range is between 1 and 100, use PERCENT100.

Choose between PERCENT100 and PCTGAUGE100 based on weighting. Event-type data is not weighted, so there is no difference. Interval-type data is not weighted if you use PCTGAUGE100 and is weighted if you use PERCENT100.

RATE Events per second

Default Type: Numeric

Default Length: 8

Default Format: BEST12.2

Default Interval Statistics: Aw

Default Event Statistics: csA

STRING — Group of printable characters
- Default Type: Character
- Default Length: 200
- Default Format: .
- Default Interval Statistics: -
- Default Event Statistics: -

SUM — Numeric sum of the values
- Default Type: Numeric
- Default Length: 8
- Default Format: BEST12.2
- Default Interval Statistics: S
- Default Event Statistics: S

TIME — SAS time value (number of seconds since midnight). TIME is treated in the same way as COUNT, because TIME represents an elapsed time, which is simply a count of seconds.
- Default Type: Numeric
- Default Length: 8
- Default Format: TIME12.2
- Default Interval Statistics: SA
- Default Event Statistics: cSA

TIMETICKS — Count of timeticks in hundredths of a second
- Default Type: Numeric
- Default Length: 8
- Default Format: BEST12.
- Default Interval Statistics: SA
- Default Event Statistics: cSA

UNIXTIME — Number of seconds since 01Jan70
- Default Type: Numeric
- Default Length: 8
- Default Format: DATETIME21.2
- Default Interval Statistics: -
- Default Event Statistics: -

YNFLAG — A single character, either Y or N
- Default Type: Character
- Default Length: 1
- Default Format: $CHAR1.
- Default Interval Statistics: -
- Default Event Statistics: -

Statistic Codes for Variables

The meanings of the statistics codes are as follows:

-	No default
A	Average
C, c	Count
D	Standard deviation
h	Sum(DURATION*rate**2)
K	Variance
M	Number of observations for which this metric has a missing value
N, n	Minimum
P	Sum(count**2/DURATION)
Q, q	Uncorrected sum of squares
S, s	Sum
V	Coefficient of variance
w	Weighted sum
X, x	Maximum
Y	Range

Lowercase characters indicate that the statistic is hidden. Hidden statistics might be required in order to calculate other requested statistics. For example, if you request the average, then the sum and the count must be kept so that you can calculate the average.

As listed above, the *w* indicates that the variable is weighted. You can also set alternate weight variables by using the WEIGHTVAR= parameter on the %CPDDUTL CREATE VARIABLE or UPDATE VARIABLE control statements or by using the Create or Edit Variable Definition window within the SAS IT Resource Management GUI.

If you do not specify the WEIGHTVAR= parameter or do not select a weight variable within the interface, then the default variable weighting and normalization are used.

Note: The names of variables in the summary levels are created by adding these same codes as the final character of the variable name. For example, if you requested sum and minimum for variable LU62IOC, then those statistics would be named LU62IOCS and LU62IOCN, respectively. Variable names that are shorter than seven characters are padded with underscores; for example, the sum for variable SMFQIO would be SMFQIO_S. Blank is used for Average instead of A; for example, variable BATCPU would be BATCPU instead of BATCPU_A. The names are not case sensitive. △

Statistical Notes

By default, analysis variables in INTERVAL tables are weighted and normalized by the DURATION variable during report-time summarization, such as the summarization when Summary Time period is not AS IS. By default, analysis variables in EVENT tables are not weighted or normalized.

In an INTERVAL table, statistical calculations for the following variable interpretation types are normalized or weighted (that is, converted to rates for comparison purposes): COUNT, TIME, TIMETICKS. These variables are normalized (or weighted) with respect to the DURATION of the interval. This is because the calculation of an Average is based on an implied “per what.”

For EVENT data this would be per job, per transaction, and so on. For INTERVAL data, the obvious choice of “per” interval might not be appropriate, because the interval

in the data can vary greatly among collectors. Other collectors might record data in fixed intervals, but the user can change the interval. In these cases the intervals are not the same, so it would be misleading to compare or analyze them as if they were the same.

By normalizing COUNT, TIME, and TIMETICKS values by the duration of the interval, we arrive at "per second" values. This results in rates for COUNT values (for example, page faults per second) and utilizations for TIME and TIMETICKS data (for example, CPU seconds per second).

If a weighted average is requested for variables with an interpretation type other than COUNT, TIME, or TIMETICKS in an INTERVAL table, then the statistical calculations are weighted by the duration of the interval.

If neither of the previous items applies, such as for variables in an EVENT table, then the statistical calculations are not weighted.

The following variable interpretation types are not converted if the table is an INTERVAL table: GAUGE, MAXIMUM, MINIMUM, INT. All other numeric variable interpretation types are weighted if the table is an INTERVAL table.

By default, no weighting or normalization occurs for EVENT tables unless the WEIGHTVAR= parameter is used to specify a weight variable. If a weight variable is specified for a variable in an EVENT table, then the same normalization or weighting rules that are described above are used.

You can set alternate weight variables by using the WEIGHTVAR= parameter on the %CPDDUTL CREATE VARIABLE or UPDATE VARIABLE control statements or by using the Create or Edit Variable Definition window within the SAS IT Resource Management GUI.

Interpretation Type Examples

Example 1: Interpretation Types for Numeric Data Type

COUNT	count of events
TIME	SAS time value (number of seconds elapsed or since midnight)
DATETIME	SAS date time value (number of seconds since 1Jan1960)
GAUGE	instantaneous reading
PERCENT	a ratio (value between 0 and 1).

As an example of the effect of interpretation type on format, suppose you have five variables, one of each of the above interpretation types. Suppose each variable contains the value 1 internally. If you display all five, they appear like this by default:

COUNT	is displayed as 1
TIME	is displayed as 0:00:01.00
DATETIME	is displayed as 1JAN1960:00:00:01.00
GAUGE	is displayed as 1
PERCENT	is displayed as 100%

Example 2: Normalization Example

Table A5.1 Normalization: Table of Observations

Observations	ABNPC	DURATION
observation 1	5	3600
observation 2	4	3600
observation 3	4	1800
observation 4	5	1800

Variable **ABNPC** is the number of abandoned calls. Variable **DURATION** is the length of the interval in seconds.

Suppose the values of **ABNPC** represent snapshots of the number of phone lines that are clearing an abandoned call *at the beginning of the observation's interval*. The appropriate average of **ABNPC**, as **GAUGE**, would be

$$\frac{5+4+4+5}{4} = 4.5$$

Suppose the values of **ABNPC** represent an accumulated count of the number of abandoned calls that occurred during the observation's interval. The appropriate average of **ABNPC**, as **COUNT**, would be

$$\frac{5calls + 4calls + 4calls + 5calls}{1hour + 1hour + \frac{1}{2}hour + \frac{1}{2}hour} = 6calls/hour = .0016666calls/sec$$

Note: **COUNT** is normalized and **GAUGE** is not. △

Example 3: Weighting Example

Table A5.2 Weighting: Table of Observations

Observations	PCTUTIL	DURATION
observation 1	70%	3600
observation 2	70%	3600
observation 3	80%	1800
observation 4	80%	1800

The variable **PCTUTIL** is percent utilization. The variable **DURATION** is the length of the interval in seconds. Suppose the values of **PCTUTIL** represent snapshots. Then the appropriate average, as **PCTGAUGE**, would be

$$\frac{70\% + 70\% + 80\% + 80\%}{4} = 75\%$$

Suppose the values of **PCTUTIL** represent the overall use of the trunks during the interval. Then the appropriate average, as **PERCENT**, would be

$$\frac{70\% \times 1hour + 70\% \times 1hour + 80\% \times \frac{1}{2}hour + 80\% \times \frac{1}{2}hour}{1hour + 1hour + \frac{1}{2}hour + \frac{1}{2}hour} = 73.3\%$$

PERCENT is weighted and **PCTGAUGE** is not.

Variable Interpretation Type Summary Table

The following table lists each of the interpretation types with its default variable type, default length, default format, and default statistics for the variable when it is used in an interval or event table. For a complete description of each interpretation type and for information on the statistical codes and note on statistical weighting of variables, refer to "Variable Interpretation Types" on page 697 and "Statistic Codes for Variables" on page 706.

Table A5.3 Interpretation Type Summary Table of Defaults

Interpretation	Type	Length	Interval Stats	Event Stats	Format
AVERAGE	Num	8	Aw	csA	best12.2
C2RATE	Num	8	Aw		best12.2
COUNT	Num	8	SA	cSA	best12.
COUNTER	Num	8			best12.
DATE	Num	8			date9.
DATETIME	Num	8			datetime21.2
DECADDRESS	Char	6			.
D2RATE	Num	8	Aw		best12.2
ENUM	Num	4			5.
FLOAT	Num	8			best12.2
FORMULA	N/A	8	N/A	N/A	
GAUGE	Num	6	csA	csA	best12.2
HEXFLAGS	Char	200			$hex.
INT	Num	6			best12.
IPADDRESS	Char	15			.
LABEL	Char	40			.
MAXIMUM	Num	8	X	X	best12.2
MINIMUM	Num	8	N	N	best12.2
NAME	Char	64			.
NETADDRESS	Char	11			.
PCTGAUGE	Num	6	csA	csA	percent9.2

Interpretation	Type	Length	Interval Stats	Event Stats	Format
PCTGAUGE100	Num	6	csA	csA	6.2
PERCENT	Num	6	Aw	csA	percent9.2
PERCENT100	Num	6	Aw	csA	6.2
RATE	Num	8	Aw	csA	best12.2
STRING	Char	200			.
SUM	Num	8	S	S	best12.2
TIME	Num	8	SA	cSA	time12.2
TIMETICKS	Num	8	SA	cSA	best12.
UNIXTIME	Num	8			datetime21.2
YNFLAG	Char	1			$char1.

APPENDIX

6

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *Getting Started with SAS IT Resource Management*
- *SAS IT Resource Management: User's Guide*
- *SAS IT Resource Management: Administration and Batch Reporting Course Notes*
- *SAS IT Resource Mangagement: Interactive Reporting Course Notes*

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Index

Numbers

A

B

C

D

E

F

G

H

I

Q

R

S

T

U

V

W

X

Y

Your Turn

If you have comments or suggestions about *SAS IT Resource Management 2.7: Macro Reference*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**