

SAP SECURITY REQUIREMENT SEC-256 (DELETION OF PERSONAL DATA)

DEVELOPMENT HOW-TO INFORMATION



END OF PURPOSE CHECK FOR APPLICATIONS CONSUMING SAP BUSINESS PARTNER

Released for SAP Customers and Partners

Valid As of Release

SAP ERP 6.0 EhP7 SP05 and SAP CRM EHP3 SP05 based on SAP NetWeaver 7.40





Document Version

1.0 (August 2014)

Target Group

SAP partner or customer solutions developers in charge of compliance for security requirement SEC-256
"SAP software shall support deletion of personal data (including personally identifiable information)"

TYPOGRAPHIC CONVENTIONS

Type style / icon	Description
<i>Quotation</i>	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Emphasis	Emphasized words or phrases in body text, graphic titles, and table titles.
All names	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
User entry	User entry texts are words or characters that you enter in the system exactly as they appear in the documentation.
<Variable user entry>	Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
KEY	Keys on the keyboard, for example, F2 or ENTER.
	Note
	Caution
	Example
	Recommendation

DOCUMENT HISTORY

Document version	Date	Description
1.0	August 2014	

TABLE OF CONTENTS

- 1 Introduction 6
- 2 Overview about the End of Purpose (Blocking) Process 6
 - 2.1 Information Lifecycle Model..... 6
 - 2.2 END OF PURPOSE (BLOCKING) PROCESS..... 7
 - 2.3 Key information about available master data blocking functionality..... 8
 - 2.3.1 Central Business Partner (cBP) 8
 - 2.3.2 ERP Customer and Vendor 8
 - 2.3.1 SCM Location 9
 - 2.4 Features of the Blocking Reports 9
 - 2.4.1 Central Business Partner Data (cBP) 9
 - 2.4.2 ERP Customer/Vendor 10
 - 2.4.3 SCM Location 10
 - 2.5 Handling of blocked master data in dependent applications 11
 - 2.5.1 Central Business Partner Data (cBP) 11
 - 2.5.2 ERP Customer/Vendor 11
 - 2.5.3 SCM Location 12
- 3 How-To of an EoP Check Implementation 12
 - 3.1 Overview about Maintenance of Application Specific Residence and Retention Periods. 12
 - 3.2 Purpose of application names 13
 - 3.3 Purpose of application rule variants 13
 - 3.4 Requirements for the EoP Check Implementation 13
 - 3.5 Step 1 – Collect Condition Field Values 14
 - 3.6 Step 2 – Map the Condition Field Values to the Application Rule Variants..... 14
 - 3.7 Step 3 - Calculate the End of Residence Date 15
 - 3.8 Step 4 - Return new SoRT data 16
 - 3.9 Mapping of Condition Fields to Application Rule Variants using ILM Rule Groups 16
 - 3.10 Store SoRT Information before the central EoP check run 17
- 4 EoP Check for Central Business Partner Data (cBP)..... 18
 - 4.1 Registration of Applications 18
 - 4.2 The “End-Of-Purpose” Check Interface..... 19
 - 4.3 The Unblocking Check Interface..... 21
 - 4.4 Enterprise Service Methods..... 22
 - 4.4.1 EnD of Purpose Check for Central Business Partner 23
 - 4.4.2 Block Notification for Central Business Partner 24
 - 4.4.3 Interim Check Results Notification for Central Business Partner..... 24

4.4.4	Unblock check for Central Business Partner.....	24
4.4.5	Unblock notification for Central Business Partner	25
4.5	Business Add-Ins (BAdIs).....	25
4.5.1	Business Partner Key Mapping	25
4.5.2	Restriction of Business Partner from EoP Check.....	25
4.5.3	Export SoRT Details of Blocked/Unblocked BP to FS Memory.....	26
4.6	Adaption by Dependent Applications.....	27
5	EoP Check for the ERP Customer and Vendor.....	28
5.1	Registration of Applications	28
5.2	The “End-Of-Purpose” Check Interface.....	29
5.2.1	Interface Methods.....	29
5.2.2	Interface Attributes.....	30
5.2.3	Initialize and Finalize methods	31
5.2.4	EOP Check Method CHECK_PARTNERS.....	32
5.2.5	Blocking method PARTNERS_PURCMPL_EXPORT.....	34
5.2.6	Blocking Process flow.....	34
5.3	The Unblocking Check Interface.....	35
5.3.1	Initialize and Finalize methods for Unblocking.....	35
5.3.2	Unblocking Methods	36
5.3.3	Unblocking process flow.....	37
5.4	Special Scenario: One Time Accounts	37
5.4.1	EOP Check Method CHECK_CPD_PARTNERS.....	38
5.5	Enterprise Service Methods.....	38
5.5.1	End of Purpose check for customer/supplier (vendor).....	40
5.5.2	Block Notification for Customer/Supplier	40
5.5.3	Interim Check Results Notification for Customer/Supplier.....	41
5.5.4	Unblock check for Customer/Supplier.....	41
5.5.5	Unblock notification for Customer/Supplier.....	41
5.6	Business Add-Ins (BAdIs).....	41
5.6.1	Mapping of Master Data ID for EoP and Unblocking	41
5.6.2	EoP Check Report - Modify Selection Before and After DB Selection	42
5.6.3	Custom Change Documents Handling for EoP Check.....	44
5.7	Adaption by Dependent Applications.....	44
5.7.1	Adaptation of Read Function Modules	45
5.7.2	Adaptation of Check Function Modules	46
5.7.3	Adaptation of Search Function Modules.....	46
5.7.4	Adaptation of BAPI Methods/Function Modules	47

5.7.5	Adaptation of Enterprise Services.....	47
6	EoP Check for the SCM Location.....	48
6.1	Registration of Applications.....	48
6.2	The “End-Of-Purpose” Check Interface.....	48
6.2.1	Interface Methods.....	49
6.2.2	Interface Attributes.....	49
6.2.3	Initialize and Finalize methods	50
6.2.4	EOP Check Method CHECK_PARTNERS.....	51
6.2.5	Blocking method PARTNERS_PURCMPL_EXPORT.....	53
6.2.6	Blocking Process flow.....	53
6.3	The Unblocking Check Interface.....	53
6.3.1	Initialize and Finalize methods for Unblocking.....	54
6.3.2	Unblocking Methods	55
6.3.3	Unblocking process flow.....	56
6.4	Adaption by Dependent Applications.....	56
7	EoP related Customizing Settings	57
7.1	Available Customizing Settings (IMG and ILM).....	57
7.2	IMG settings - Central Business Partner Data (cBP)	57
7.3	IMG settings – ERP Customer and Vendor.....	58
7.4	IMG settings – SCM Location.....	58
7.5	ILM settings.....	58
7.5.1	IRM_CUST (delivered by SAP) for Master Data ILM objects (CA_BUPA, FI_ACCRECV, FI_ACCPAYB, FI_ACCKNVK, SCMB_LOC)	58
7.5.2	IRM_CUST for ILM objects provided by the Business Partner Dependent Application 58	
7.5.3	IRMPOL (To be maintained by SAP customers).....	59
7.5.4	IRM_CUST_CSS (To be maintained by SAP customers).....	59
8	Integration Scenarios.....	59
8.1	Integration of the EoP check for Central Business Partner and the ERP Customer/Vendor 59	
8.2	Distribution of the Blocking Information.....	60
8.2.1	Central Business Partner Data (cBP)	60
8.2.2	ERP Customer/Vendor	60
8.2.1	SCM Location	60
9	Copyright and Disclaimer	61

1 INTRODUCTION

This document provides for partner or customer solutions (ABAP or non-ABAP based) a technical overview about the available business partner master data blocking functionality and the interfaces to be considered for the creation of an End of Purpose check for an own application.

In addition it should provide information about the functionality, which can be used to do the necessary adaptations in the corresponding functionality to handle correctly blocked business partners.

2 OVERVIEW ABOUT THE END OF PURPOSE (BLOCKING) PROCESS

2.1 INFORMATION LIFECYCLE MODEL

The lifecycle of business data comprises certain phases such as data usage, residence time and retention time. Figures 2-1 as example for blocking and 2-2 as example for destruction show how these phases interrelate. Data usage starts with the creation of data and ends with the date when the business purpose and the residence time are over. When the business is complete – that means that end of business is reached - the residence and retention periods start.

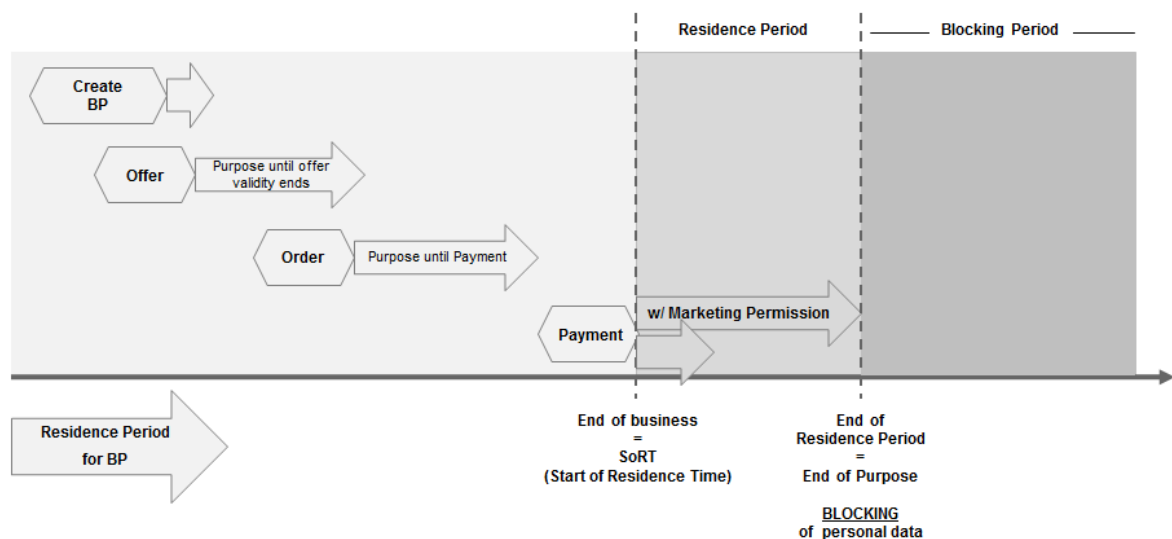


Figure 2-1 Information life cycle phases for residence periods and important events for blocking such as start of residence time (SoRT), end of purpose (EoP) and end of residence time (EoRT)

As shown in Figure 2-1 can each business document have defined its own residence periods with reference to a business partner. During the residence time data remains in the database and can be used in case of subsequent processes such as returns, warranty issues or even new business. This time period can take for example several months. The end of purpose for a business partner is reached when longest residence time of all related business documents is over. After the residence time, expired data have to be blocked in a way that regular users and processes cannot access this data any more, but only authorized users such as data privacy officers or auditors.

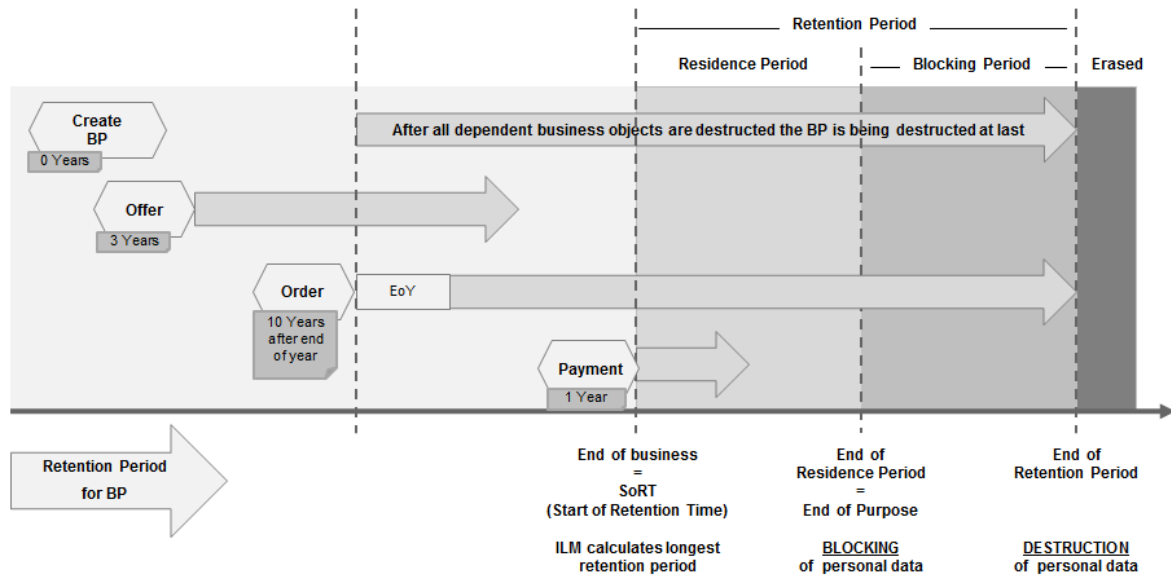


Figure 2-2 Information life cycle phases for retention periods and important events for destruction such as start of retention time (SoRT), end of purpose (EoP) and end of retention time (EoRT)

As shown in Figure 2-2 starts the retention time also with end of business and ends according to legal requirements after a certain time period such as 7 or 10 years. The purpose to store business partner data exists only due to their usage in related business documents. So usually no retention periods can be defined for the business partner master data itself. Instead the retention period is defined by the retention periods of the related business documents. After the longest retention time of the related business documents has expired, the business partner master data has to be destroyed. So it is ensured that the business partner data remain available until the last related business document data that uses a particular business partner is destructed.

2.2 END OF PURPOSE (BLOCKING) PROCESS

An end of purpose check determines whether data is still relevant for business activities based on the retention period defined for the data. The retention period of data consists of the following phases:

- Phase one: The relevant data is actively used.
- Phase two: The relevant data is actively available in the system.
- Phase three: The relevant data needs to be retained for other reasons.

For example, processing of data is no longer required for the primary business purpose, but to comply with legal rules for retention, the data must still be available. In phase three, the relevant data is blocked.

Blocking of data prevents the business users of SAP applications from displaying and using data that may include personal data and is no longer relevant for business activities.

Blocking of data can impact system behaviour in the following ways:

- Display: The system does not display blocked data.
- Change: It is not possible to change a business object that contains blocked data.
- Create: It is not possible to create a business object that contains blocked data.
- Copy/Follow-Up: It is not possible to copy a business object or perform follow-up activities for a business object that contains blocked data.
- Search: It is not possible to search for blocked data or to search for a business object using blocked data in the search criteria.

It is possible to display blocked data if a user has special authorization; however, it is still not possible to create, change, copy, or perform follow-up activities on blocked data.

2.3 KEY INFORMATION ABOUT AVAILABLE MASTER DATA BLOCKING FUNCTIONALITY

2.3.1 CENTRAL BUSINESS PARTNER (CBP)

Blocking Indicator	Field XPCPT in database table BUT000
Table for SoRT info	BUTSORT
Business Function	BUPA_ILM_BF
ILM Object	CA_BUPA
Blocking Report Transaction	BUPA_PREPARE_EOP BUPA_PRE_EOP
Unblocking Request Transaction	BUPA_REQ_UNBLOCK BUP_REQ_UNBLK
Unblocking Report Transaction	BUPA_PREPARE_EOP BUPA_PRE_EOP

2.3.2 ERP CUSTOMER AND VENDOR

Blocking Indicator	Field CVP_XBLCK* in database tables LFA1, LFB1, KNA1, KNB1, KNVV, KNVK
Table for SoRT info	CVP_SORT
Business Function	ERP_CVP_ILM_1
ILM Objects	FI_ACCRECV, FI_ACCPAYB, FI_ACCKNVK
Blocking Report Transaction	CVP_PREPARE_EOP CVP_PRE_EOP
Unblocking Request Transaction	BUPA_REQ_UNBLOCK BUP_REQ_UNBLK

Unblocking Report Transaction	CVP_UNBLOCK_MD CVP_UNBLOCK_MD
--------------------------------------	----------------------------------

2.3.1 SCM LOCATION

Blocking Indicator	Field LOC_XBLCK in database table /SAPAPO/LOC for block on general level; existence of record in database table /SCMB/LOC_BLOCK for block on company code level
Table for SoRT Info	/SCMB/LOC_SORT
Business Function	SCM_SCMB_LOC_ILM_1
ILM Objects	/SCMB/LOC
Blocking Report Transaction	/SCMB/RLOC_PREPARE_EOP /SCMB/LOC_PRE_EOP
Unblocking Report Transaction	/SCMB/RLOC_UNBLOCK_MD /SCMB/LOC_UNBLOCK_MD

2.4 FEATURES OF THE BLOCKING REPORTS

2.4.1 CENTRAL BUSINESS PARTNER DATA (CBP)

- Prerequisites:
To use the report, the Data Protection Administrator role have to be assigned to the user. (activity 05 “Lock” of authority object B_BUP_PCPT “Business Partner: Purpose Completed”)
- Execution Modes
 - Check end of purpose: The report checks to see if the business partner can be blocked.
 - Reset end of purpose: The report resets the blocking status of the business partner.
- Further Selection Criteria
 - Interim check: The report performs a local check, where only the business partners in that system are checked. This check does not set the completion flag or blocking status.
 - Overall check: The report checks all applications and their function modules connected to the master system. The completion flag can be set when all the applications confirm that the business purpose of the business partner is completed.
 - Consider check date: The business partners that are in use until a certain date are excluded.
 - Consider interim results: The business partners for which interim check is not executed are excluded.
 - Check all application for EoP: The report checks end of purpose function module for all the applications.
- Parallel Processing

- Block Size: Specifies the number of business partners to be processed at a time
- Max Processes: Specifies the number of parallel processes that is created to process the blocking request.
- Server Group: Specifies the server group where the blocking request must be processed.
- Control
 - With application log: The application log is displayed.
 - With detail application log: The detailed application log is displayed.
 - Save application log: The application log is saved.
 - Test run, check only: The report is run in test mode, and no data is stored in the table.

2.4.2 ERP CUSTOMER/VENDOR

The ERP customer/vendor blocking report provides similar functionality as the central business partner blocking report.

Differences:

- Support execution only for customer or for vendor master data (but not for both at the same time)
- Offers selection of data to be processed (similar to the existing archiving functionality) for the data levels, on which separate blocking is possible:
 - “AL – Check on all Data Levels” EoP check is started on general level, so complete Customers or Vendors are checked and can be blocked.
 - “FI – Check in Company Code Level” -> EoP check is started on company code level, so only company code specific data of Customers (KNB1) or Vendors (LFB1) are checked and can be blocked.
 - “CP – Check only Contact Persons” -> EoP check is started for contact persons assigned to the selected Customers or Vendors, only contact persons (KNVK) are checked and can be blocked.
- Default settings for parallelization can be configured in the customizing (view CVP_C_PARALLEL_V)

2.4.3 SCM LOCATION

- Prerequisite:

To use the report, the Data Protection Administrator role has to be assigned to the user. (Activity 05 “Lock” of authority object /SCMB/LOCB “Location: Access based on Authorization Group”)
- Supports execution for the following location types: Customer, Vendor, Transportation Service Provider and Subcontractor
- Supports blocking on two separate data levels:
 - *All Data Levels* – this is also referred as blocking on general level. Complete location is checked and blocked
 - *Company Code Level* – EoP check is started on company code level. After blocking, no business bound with the location and plants of the company code is possible. Blocking on company code level is possible only for locations integrated from ERP (i.e. only possible with processing option *Test Mode – ERP locations* for report run in SCM or in *Overall Check (Remote) w/ Setting Compl. Flag* execution mode if the report is run in ERP)
- Options
 - *Skip Subsequent EoP Checks If Ongoing Business Found* – If an application reports ongoing business for a location, report skips EoP check of remaining applications for the location

- *Consider Next Check Date* – Report skips EoP check for locations where next check date is in future.
- Parallel Processing
 - *Number of Objects per Process* – Indicates the number of master data that the system processes in one block
 - *Max. Number of Parallel Processes* – Specifies the number of parallel processes that are created to process the blocking request
 - *Server Group* – Specifies the server group for which the blocking request must be processed
 - Default setting for parallelization can be configured in the customizing (view /SCMB/LOCPARAL_V)
- Processing Options
 - *Test Mode – SCM locations* – Testing mode for locally created SCM locations. This mode will filter out integrated locations. This mode is used to simulate what happens when using Production mode.
 - *Test Mode - ERP locations* – Testing mode for SCM locations created via integration. This mode will filter out locally created locations. This mode is used to simulate what happens in SCM when you run EoP check for customer/vendor in ERP system.
 - *Production Mode* – With Production mode, locally created SCM locations are processed and blocked if possible

2.5 HANDLING OF BLOCKED MASTER DATA IN DEPENDENT APPLICATIONS

2.5.1 CENTRAL BUSINESS PARTNER DATA (CBP)

- Requirement for read authority:
To see blocked master data, read authority (activity 03 “Display” of authority object B_BUP_PCPT “Business Partner: Purpose Completed”) have to be assigned to the user.
- How to determine, if a BP is blocked?
 - Retrieve pure blocking information:
Function Module BUPA_XPCPT_GET (RFC enabled)
 - Including authority check:
Function Module BUPA_DP_AUTHCHECK
Function Module BUPA_DP_AUTHCHECK_API
Function Module BUPA_DP_AUTHCHECK_MASS
- How is this covered in the central read APIs?
 - All central read APIs include now the check on the block indicators, the necessary authority check and to return no or initial data for the unchanged calls of the APIs.

2.5.2 ERP CUSTOMER/VENDOR

- Requirement for read authority:
To see blocked master data, read authority (activity 03 “Display”) for the customized authorization group (BEGRU) for the following authorization objects have to be assigned to the user:
F_KNA1_BED

F_BKPF_BEK
 F_BKPF_BED
 V_KNA1_BRG
 F_KNKK_BED

In addition activity 03 “Display” of authority object B_BUP_PCPT “Business Partner: Purpose Completed” is required due to the read access for blocked address data.

- How to determine, if a customer/vendor/contact person is blocked?
 - To determine the blocking status of a customer / vendor / contact person:
Function Modules CVP_GET_BLOCK_STATUS_CUST/VEND/CONT
 - To return also the user authorization for a customer / vendor / contact person:
Function Modules CVP_AUTHCHECK_CUST*
Function Modules CVP_AUTHCHECK_VEND*
Function Modules CVP_AUTHCHECK_CONT*
- How is this covered in the central read APIs?
 - All central read APIs include now the check on the block indicators, the necessary authority check and to return no or partly initial data for the unchanged calls of the APIs.

2.5.3 SCM LOCATION

- Read authority requirement
To see blocked master data, read authority (activity 03 *Display* of authority object /SCMB/LOCB *Location: Access based on Authorization Group*) must be assigned to the user.
Activity 03 *Display* of authority object B_BUP_PCPT *Business Partner: Purpose Completed* is required due to the read access for blocked address data.
- SCM Location blocking
 - To determine the blocking status and authorization for SCM Location use function module /SCMB/LOC_GET_BLOCK_STATUS_LOC
- Read APIs
 - Read APIs for SCM locations include new exporting table ET_LOC_BLOCK_STATUS, where locations with blocking flag are listed. By default, fields with personal data are returned as initial if user doesn't have special authorization

3 HOW-TO OF AN EOP CHECK IMPLEMENTATION

One of the central aspects for the blocking concepts of the central business partner respectively ERP Customer and Vendor or SCM Location is, that application specific retention periods have to be maintained twice. Of course this has to be done in transaction IRMPOL for the ILM object of the application. But also for the ILM object of the assigned master data object.

The point here is that for the master data object not the same condition fields as in the application are available. So the application rule variant concept was introduced to provide a mapping possibility for customers to maintain the same retention periods in the master data ILM object as for the application object.

3.1 OVERVIEW ABOUT MAINTENANCE OF APPLICATION SPECIFIC RESIDENCE AND RETENTION PERIODS

	Master Data ILM Objects	Dependent Application Object
--	--------------------------------	-------------------------------------

Residence Periods	<p>Residence periods are maintained for the master data ILM objects with reference to:</p> <p>The master data itself, e.g. if new master data without usage shouldn't be blocked too early For each dependent application object maintained via the application name/application rule variant concept to consider these periods according to the usage of a master data object ID before blocking</p>	<p>Blocking and maintenance of residence periods for the ILM object of a dependent application object are not required.</p>
Retention Periods	<p>Retention periods are maintained with reference to:</p> <p>The master data itself, e.g. if master data without usage shouldn't be destroyed too early. For each dependent application object maintained via the application name/application rule variant concept to consider the application specific retention period before destruction of a particular master data object ID.</p>	<p>Retention periods are maintained for the ILM object of a dependent application object to ensure destruction of application data only after the end of the retention period.</p>

3.2 PURPOSE OF APPLICATION NAMES

For the mapping of dependent applications to the relevant master data object at first the application name is provided, which groups similar application objects. It is important to know, which ILM objects of the application are relevant for which application name. The purpose of this is to get an understanding about, for which application names the (double) maintenance of retention policies and residence policies is necessary.

3.3 PURPOSE OF APPLICATION RULE VARIANTS

Application specific retention policies in the master data ILM objects, which are maintained only for the application name, allow only the definition of exactly one retention period. But depending on the scope of the application name and the related ILM objects, there can be the requirement to define several application object specific retention periods. And of course it can be required to maintain several retention periods for a single application object, especially if retention periods from several countries have to be considered. The application specific ILM objects offers several condition fields for this purpose, which are all not available in the master data ILM objects.

So the application rule variant is provided as an additional condition field, which serves as a mapping field to differentiate several required retention periods of an application name.

3.4 REQUIREMENTS FOR THE EOP CHECK IMPLEMENTATION

A dependent application, which needs to support the EoP check of the cBP or ERP customer or vendor or SCM location, requires further information to find out the correct values for the SoRT information to be provided to the master data.

The tasks to be fulfilled by the EoP check implementation are the following:

1. Find existing data in the own application objects with relation to a given list of master data object IDs and collect per related application ILM object all values of the condition fields.
2. Map the condition field values to the application rule variants, which are maintained eventually for the own application name.
3. Call the ILM method to calculate the end of residence date for each determined combination of application name and application rule variant
4. Return new SoRT data for each determined combination of application name and application rule variant

3.5 STEP 1 – COLLECT CONDITION FIELD VALUES

The registered EoP check functionality of an application is called for the given application name and a list of master data object IDs. During the creation of the EoP check it has to be defined, which application objects and which related ILM objects are in scope for the implementation and the application name. If relevant the ILM objects to be considered can be also determined at the beginning of the EoP check via a call of method `CL_LRM_BS_RULE_EXEC=> GET_OBJECT_TYPE_FOR_BOR_OBJ` by specifying the name of the business (BOR) object.

For these ILM objects now existing data for the given master data object IDs have to be selected from the corresponding database tables. How to consider already archived application data is described in chapter “Store SoRT Information before the central EoP check run”.

In addition the relevant start of retention date as aggregated result (what means usually the highest date) of all relevant application-specific time references has to be determined. And also the application specific status checks for end of business need to be done.

If no business at all is found for a master data object ID, then this information has to be returned for the application name as described in step 4.

Consider all Data??

If data with ongoing business is found, then it should be checked if only this data is to be considered in the next steps. It adds no benefit, if in this case also the full end of residence time calculation for data in status End of Business is done. But it can improve the performance a lot, if only the really required data is processed.

3.6 STEP 2 – MAP THE CONDITION FIELD VALUES TO THE APPLICATION RULE VARIANTS

The task in this step is to find the application rule variant values of the own application name, which are assigned to the found condition fields determined within step 1.

Important Remark:

The simple case is a 1:1 relation between valid ILM retention rule, existing application data and application rule variant. But even for a single application object there can be several valid application rule variants, just depending on the available data and the different relevant ILM retention rules. So the EoP check implementation needs to take care of the possibility, that several application rule variants are found, which are all valid in parallel. In addition also no application rule variant can be found, which means that the processing has to continue just with the application name and an initial application rule variant value. A basic method to find the correct application rule variants in regard to the collected condition field values is to create an application specific mapping customizing. This customizing should link all possible condition fields to an application rule variant value. This is very application specific and results in one customizing per relevant ILM object.

As alternative to this concept the ILM rule group concept is available. This concept uses the already maintained valid mapping as maintained per ILM object of an application in transaction `IRMPOL` for each required/defined retention rule. These retention rules can be linked to an ILM rule group value. An ILM rule

group can be defined in transaction `IRM_CUST_CSS` and stores the time period information, which is to be defined per rule. So it can be already used to group equal retention periods per application. The customizing for application rule variants offers a mapping to an ILM rule group value as a 1:1 relation. So it is possible via the determination of assigned ILM rule groups to find, which application rule variants have to be used for the following steps of the EoP check implementation. The following picture shows as example the customizing for the maintenance of application rule variants for a customer / vendor and the assignment of ILM rule groups.

Change View "Define / Store Application Rule Variants for EoP check":					
Define / Store Application Rule Variants for EoP check					
ID Type	Appl Name	Application Description	Application R...	Regelgr.	Application Rule Variant Description
1 Customer master data	ERP_FI	Financial Accounting	0001_AA	ERP_FI_GR2	CCod 0001 and DocType AA
1 Customer master data	ERP_FI	Financial Accounting	0001_DR	ERP_FI_GR1	CCod 0001 and DocType DR

Figure 3-1 Example for the customizing of application rule variants for a customer / vendor and the assignment of ILM rule groups

The details for how to implement this step using ILM rule groups will be explained in chapter "Mapping of Condition Fields to Application Rule Variants using ILM Rule Groups".

The way for how to find the assigned application rule variants for the own application name is depending on the master data object:

- For the cBP the function module `BUPA_BUTEOPARV_GET` can be called with import parameters `IV_APPL_NAME` (the application name) and `IT_APPL_RULE_GROUP` (the determined list of ILM rule groups).
- For the ERP Customer/Vendor the method `GET_BY_RULE_GROUPS` of class `CVP_CL_EOPARV` can be called. Importing parameters are `IV_ID_TYPE` (Customer, Vendor or Contact Person), `IV_APPL_NAME` (the registered application name) and `IT_RULE_GROUP` (the determined list of ILM rule groups).
- For SCM Location, you can call method `GET_BY_RULE_GROUPS` of class `/SCMB/LOC_CL_EOPARV`. The import parameters are `IV_ID_TYPE` (customer, vendor, transportation service provider or subcontractor), `IV_APPL_NAME` (the registered application name), and `IT_RULE_GROUP` (the determined list of ILM rule groups).

3.7 STEP 3 - CALCULATE THE END OF RESIDENCE DATE

The result of step 2 is a list of application rule variants, which are relevant per master object ID.

Now it is important to separate the list regarding the status "End of Business". The following calculation for the end of residence period is not required in case of ongoing business. But the information about ongoing business has to be returned per application name and application rule variant to the calling blocking report as described in step 4.

For each combination of application name and application rule variant now the end of residence date has to be separately calculated using method `GET_RESIDENCE_RULE_F_VALUES` of class `CL_LRM_BS_RST_RULE_EXEC`.

For the ILM objects of the business partner master data objects different residence periods can be maintained. It is defined, that the policy type which needs to be used is the audit area name "BUPA_DP". This is always specified in the method call together with the instance origin containing the current system ID and client. Important is to fill all condition values into the source fields table. This includes for each entity the application name and application rule variant, but also further condition fields like the business partner type (cBP -> `BP_TYPE`) or the account group (ERP Customer/Vendor -> `KTOKD`) or the company code (ERP Customer/Vendor -> `BUKRS`) or Location Type (SCM location -> `EOP_LOC_TYPE`). Relevant are here the technical source field values to be specified together with table and field name.

Beside the condition field input values also the determined input value for the time reference parameter `START_RET_DATE` (`CVP_SORT-ST_RET_DATE` or `BUTSORT-ST_RET_DATE` or `/SCMB/LOC_SORT-ST_RET_DATE`) have to be added.

3.8 STEP 4 - RETURN NEW SORT DATA

The date value which is returned in the variable `LV_RES_TIME_END` as result of the method call in step 3 will hold the end of residence date. This date should be compared with the current date. If the end of residence date is greater than today's date, then the purpose completion status field `PURPOSE_COMPLETION_STATUS` is to be set to "2" (Business Ongoing) and if the if the start of retention date is less than today's date, then the purpose completion status field is to be set to "3" (Business Completed).

If no data was found in step 1 for a master object ID, then the purpose completion status field has to be set to "1" (no business made with BP at all). This information should be returned for the registered application name.

This information is returned in the exporting table for the SoRT data together with the master data object ID and the business system ID for each entity of application name and application rule variant. If possible also a next check date should be set (e.g. if the end of residence date is still in future) in case of ongoing business.

Otherwise the `ST_RET_DATE` with the highest start of retention date that was determined in step 1 should be returned.

3.9 MAPPING OF CONDITION FIELDS TO APPLICATION RULE VARIANTS USING ILM RULE GROUPS

The ILM framework provides an API, which can be used to find for given condition fields as input values the assigned ILM rule groups according to the in transaction `IRMPOL` active retention policies. To be called for this purpose is the method `GET_VALID_RULES` of class `CL_LRM_API`. It has to be called per application ILM object and for each separate entity set of condition parameters.

Input parameters are:

- `IV_OBJECT_CATEGORY` to be set always with the value `"OT_FOR_BS"`
- `IV_OBJECT_TYPE` to be set with the name of the application specific ILM object
- `IV_POLICY_CATEGORY` to be set with the value `"RTP"` for retention policies
- `IT_CONDITIONS` to be filled with the field name and value of the current entity set of condition parameters. In addition always the default condition fields `CLIENT = sy-mandt` and `SYSTEM_ID = sy-sysid` have to be added.

The exporting parameter `ET_VALID_RULES` contains then information about retention rules, which matches the input parameter values. The field `RULE_GROUP` contains the information about the assigned ILM rule group. The in each call returned ILM rule groups (as several are possible!) are then the input to find the assigned application rule variants for the own application name. The details on how to do this are described in step 2 above.

One restriction of the method `GET_VALID_RULES` of class `CL_LRM_API` is that not the technical source field values together with table and field name as for method `GET_RESIDENCE_RULE_F_VALUES` of class `CL_LRM_BS_RST_RULE_EXEC` has to be used. Instead of this the "logical" field name of the condition field has to be specified. This means also, that the field value has to be determined too, especially if the indirect value determination using the method `GET_FIELDVALUES` of the BAdIs `BADI_IRM_OT_FLD` or `BADI_IRM_OC_SF` is used for the condition respectively standard condition field.

An example program for how to call method GET_VALID_RULES of class CL_LRM_API can be found in the following example program:

```
REPORT  zlrn_api_test_template.

PARAMETERS:
  p_oc TYPE lrm_object_category DEFAULT 'OT_FOR_BS',
  p_pc TYPE lrm_policy_category DEFAULT 'RTP',
  p_ot TYPE lrm_object_type DEFAULT 'FLIGHT_BOOKINGS'.

DATA:
  lt_valid_rules TYPE if_lrm_types=>ty_t_rules_with_policy.

DATA:
  lt_re_values TYPE if_lrm_re_exec=>ty_t_field_values,
  lv_carrid TYPE s_carr_id VALUE 'AA'.

FIELD-SYMBOLS:
  <lv_value> TYPE any,
  <ls_re_field_value> TYPE if_lrm_re_exec=>ty_s_field_value.

DATA:
  lt_conditions TYPE if_lrm_types=>ty_th_field_name_and_value,
  ls_condition TYPE if_lrm_types=>ty_s_field_name_and_value.

ls_condition-v_field_name = 'CARRID'.
CREATE DATA ls_condition-r_field_value TYPE s_carr_id.
ASSIGN ls_condition-r_field_value->* TO <lv_value>.
<lv_value> = lv_carrid.
INSERT ls_condition INTO TABLE lt_conditions.
CLEAR ls_condition.

ls_condition-v_field_name = 'CLIENT'.
CREATE DATA ls_condition-r_field_value TYPE sy-mandt.
ASSIGN ls_condition-r_field_value->* TO <lv_value>.
<lv_value> = sy-mandt.
INSERT ls_condition INTO TABLE lt_conditions.
CLEAR ls_condition.

ls_condition-v_field_name = 'SYSTEM_ID'.
CREATE DATA ls_condition-r_field_value TYPE sy-sysid.
ASSIGN ls_condition-r_field_value->* TO <lv_value>.
<lv_value> = sy-sysid .
INSERT ls_condition INTO TABLE lt_conditions.
CLEAR ls_condition.

CALL METHOD cl_lrm_api=>get_valid_rules
  EXPORTING
    iv_object_category = p_oc
    iv_object_type     = p_ot
    iv_policy_category = p_pc
    it_conditions      = lt_conditions
  IMPORTING
    et_valid_rules     = lt_valid_rules.
```

3.10 STORE SORT INFORMATION BEFORE THE CENTRAL EOP CHECK RUN

In step 1 of the EoP check the existing data for the given master data object IDs is selected from the corresponding database tables. For performance reasons a look up of already archived application data should be avoided, even if they also can contain SoRT information, which has to be considered for the destruction of the master data object. For this reason we recommend to store SoRT information for relevant application data also already before the central EoP check run is executed. The following options are available to implement this:

- Use of an own “shadow” database table to store SoRT information with regard to the application objects and consider these information as input during the EoP check (in step 1).
- To directly store SoRT information for the related cBP (in database table `BUTSORT`) or ERP customer or vendor (in database table `CVP_SORT`) or SCM location (in database table `/SCMB/LOC_SORT`) for the registered application name and the corresponding application rule variants.
- To use the archive info system with appropriate archive info structure within the EoP check implementation to select also already archived data.

The first and the second option can be implemented during the archiving of the application data or already before, when the end of business for the application data is known. This can have huge performance advantages during the EoP check for the application.

The disadvantage for the first solution is, that an additional database table has to be used, which aggregates already existing application data and is later only transferred into the database tables `BUTSORT` respectively `CVP_SORT` or `/SCMB/LOC_SORT`. In addition a separate deletion concept is required to delete the relevant entries after the archiving of the business partner master data.

The second option requires also the determination of the application rule variants based on the archived data as described in step 2 of the EoP check. The SoRT data (then always with purpose completion status “3” (Business Completed) and a given start of retention date) can be stored for the cBP by calling function module `BUPA_SORT_CHANGE_DETAIL`, for the ERP Customer/Vendor using the methods of class `CVP_CL_SORT_API` and for the SCM location using the methods of class `/SCMB/LOC_CL_SORT_API`. A disadvantage of this approach can be, that based on this data no reference to the application data is possible (or only with high manual effort), for which the entries have been created.

The third option requires the use of the archive info system (transaction `SARI`). Prerequisite is, that the relevant archive info structures have to be activated and filled with data before the central EoP check run. In addition the possible negative effect on the performance during the central EoP check run needs still to be analyzed.

4 EOP CHECK FOR CENTRAL BUSINESS PARTNER DATA (CBP)

The following description should provide a guide for how to implement an EoP check specifically for the cBP. The general logic is described in the previous chapter “How-To of an EoP Check Implementation”. This chapter concentrates on the cBP specific customizing, interfaces, BAdIs and supporting APIs.

4.1 REGISTRATION OF APPLICATIONS

Every consuming application of the central business partner which wishes to ILM enable the central business partner deletion process needs to first register its application name. The blocking process completely works based on the applications which have been registered.

For the registration start transaction `SPRO` and choose the following path:

Cross-Application Components -> Data Protection -> Blocking and Unblocking of Data -> Business Partner
Add the application name to the IMG entity “*Define and Store Application Names for EoP Check*”:

Define and Store Application Names for EOP Check		
Appl Name	Item	Application Description
BUB		BUSINESS PARTNERS
BUP		BUSINESS PARTNERS RELATIONSHIPS

Figure 4-1 Example for the customizing of application names of the central business partner

4.2 THE “END-OF-PURPOSE” CHECK INTERFACE

The next important step in the ILM enablement of business partner consuming applications is the creation of the end-of-purpose check function module. Every consuming application of the central business partners needs to create end-of-purpose check function module(s) (FMs) for their respective application.

For the registration of these function modules in the customizing start transaction *SPRO* and choose the following path:

Cross-Application Components -> Data Protection -> Blocking and Unblocking of Data -> Business Partner

Add an entry for each function module with its corresponding application name to the IMG entity “Define Application Function Modules Registered for EoP Check”:

Application Function Modules Registered for EoP Check			
Appl Name	Item	Function module	Reuse ILM Object Name
BKKA	0	BKKA_BUPA_EOP_CHECK	
BUB	1000000	BUB_BUPA_EOP_CHECK	
BUCP	0	BUCP_BUPA_EOP_CHECK	
BUP	2000000	BUP_BUPA_EOP_CHECK	

Figure 4-2 Example for the customizing of function modules registered for EoP check of the central business partner

The main purpose of these function modules is to calculate the residence period by integrating with ILM. These applications specific end-of-purpose check FMs would be called sequentially during the blocking process by the central blocking report and each of these FMs are expected to return the SORT information back to the blocking report, which will in-turn take a decision whether to block or otherwise. The applications consuming the business partner has to answer the question whether there is still a business purpose to store business partner data or not. If business for a certain business partner is complete from application perspective, the application has to return end-of-business status and the start-of-retention-time date. The following is the expected signature of the end-of-purpose check function module from all the consuming applications:

Parameter	Type	Associated Type	Description
IT_PARTNERS_GUID (IMPORTING)	TYPE	BUP_PARTNER_GUID_T (Not Optional)	Table Type with a list of BPs, which need to be checked for purpose completion
IT_PARTNERS_TYPE (IMPORTING)	TYPE	BUPA_PARTNER_TYPE_T	Table Type for Partner Number and Partner Type to provide necessary input information for end of residence time calculation
IV_APPL (IMPORTING)	TYPE	BU_APPL_NAME	BP Consuming Application

ET_PUR_CMPLT_PARTNERS (EXPORTING)	TYPE	BU_SORT_LOCAL_T (PARTNER PARTNER_GUID BUSINESS_SYSTEM APPL_NAME APPL_RULE_VARIANT ST_RET_DATE NEXTCHKDT PUR_CMPL_STATUS STATUS)	This parameter returns a table of BP data with purpose completion code and SoRT.
ET_MESSAGES (EXPORTING)	TYPE	BUSLOCALMSG_T (local partner id, local partner GUID, application name + Fields of structure BAPIRET2)	Message table

The end-of-purpose signature has 2 importing parameters namely, IT_PARTNERS_GUID & IV_APPL. Through the IT_PARTNERS_GUID parameter the blocking report will pass the list of business partners to be checked either in the form of a partner id and a partner GUID. The reason to pass both is to give the consuming applications the flexibility to work with either of the unique identifiers whichever is suited for the applications. Through the IV_APPL parameter the relevant application name would be passed to the end-of-purpose check module which will in-turn be needed inside the end-of-purpose check module logic.

The end-of-purpose signature has 1 exporting parameters namely, ET_PUR_CMPLT_PARTNERS. Basically the blocking report expects the details which are necessary to be stored in the central table BUTSORT for each of the business partners to be checked. Please read through the following paragraph to get know more about the individual fields of the type:

- BUSINESS_SYSTEM is the system where the end-of-purpose module has been executed for residence rule calculation, this is important in the remote run of the blocking report where the same partner's end-of-purpose needs to be calculated in a distributed environment. Usually to be returned is the logical name of the local system, which can be determined like in following example:

```
DATA: l_own_logical_system type logsys,
call function 'OWN_LOGICAL_SYSTEM_GET'
importing
  own_logical_system          = l_own_logical_system
exceptions
  own_logical_system_not_defined = 1
  others                      = 2.
if sy-subrc <> 0.
  l_own_logical_system = sy-sysid.
endif.
```

- APPLICATION_NAME is also needed to know the partner consumption information for each application. The importing parameter IV_APPL provides the relevant application name, for which the registered function module was called. Within an application if there are many sub-application for which the end-of-purpose needs to be calculated individually, the applications can make use of the "application rule variant" concept to get more granularities. For example, if there are application rule variants maintained for an application then the SORT information is needed for each rule variant of that application, that is relevant due to the application own customizing for a checked business partner.
- The purpose completion code should be returned in the field PUR_CMPL_STATUS. Application needs to fill the purpose completion code either with:
 - 1 = n/a (no business made with BP at all),
 - 2 = no (business is ongoing with BP),
 - 3 = yes (business is complete with BP and residence time is over).

- In the case of the residence date being met during the calculation by ILM, then the `START_OF_RETENTION` in the form of a date needs to be filled and returned to the blocking report together with `PUR_CMPL_STATUS = 3` (business is complete with BP and residence time is over).
- `NEXTCHKDT` – “Next Check Date” is the concept which has been introduced for optimization reasons. To put more light into this concept, imagine there 1000 partners which are checked for end-of-purpose and only 500 partners have reached end-of-purpose successfully. The remaining 500 partners imply there are still open business transactions happening by the consuming applications. The applications which return a “Business Ongoing” status through the structure field `PUR_CMPL_STATUS` for these partners have to send a date which should be greater than the current date, which would be treated as the next check date for this partner for going through the process of end-of-purpose again.

If the applications responsible for returning status “Business Ongoing” does not set the “Next Check Date” value, the central blocking report would default the next check date based on the central customizing. The defaulting by the central blocking report would be done based on the following logic:

In the blocking report once the purpose check is completed for each application, a check would happen to find whether the applications reporting ongoing business are returning a next check date. If the next check date is not returned, the report would read the value maintained in the table `BUTNXTCHK`. In this customizing a value for a time period can be maintained, which will be used to determine the default next check date starting from the current date. The calculated next check date is then stored in the `SoRT` data returned by the corresponding application. If the application returns a check date, which is larger than the calculated default next check date due to the customizing, then the check date set by the application is overwritten with the calculated default next check date to avoid too long periods until the next end of purpose check for the corresponding business partner is done.

- `STATUS` field which is present in the structure `BU_SORT_LOCAL` should not be filled by the application specific end-of-purpose check modules, this field would be filled by the central blocking report.

4.3 THE UNBLOCKING CHECK INTERFACE

Unblocking of a central business partner can be requested via transaction `BUP_REQ_UNBLK`. If the unblocking request is approved via the corresponding option in transaction `BUPA_PRE_EOP`, then an additional consistency check by the dependent applications is possible. Every consuming application of the central business partners can create an unblocking check function module for their respective application. For the registration of these function modules in the customizing start transaction `SPRO` and choose the following path:

Cross-Application Components -> Data Protection -> Blocking and Unblocking of Data -> Business Partner
 Add an entry for each function module with its corresponding application name to the IMG entity “Define Registered Function Modules for Unblock BP”:

Application Function Modules Registered for Unblock Check		
Appl Name	Item	Function module

Figure 4-3 Example for the customizing of function modules registered for unblocking check of the central business partner

These applications specific unblocking check FMs would be called sequentially after the approval of an unblocking request and before the blocking information of a business partner is removed on the database.

Expected is to return a unblocking status for each business partner, which is verified before the unblocking process continues.

The following is the expected signature of the unblocking check function module from all the consuming applications:

Parameter	Type	Associated Type	Description
IT_PARTNERS_GUID (IMPORTING)	TYPE	BUP_PARTNER_GUID_T (Not Optional)	Table Type with a list of BPs, which should be checked for unblocking
IV_APPL (IMPORTING)	TYPE	BU_APPL_NAME	BP Consuming Application
ET_UNBLK_STATUS (EXPORTING)	TYPE	BUPARTNER_UNBLK_STATUSREMOTE_T	This parameter returns a table of BP data with the unblocking check result
ET_MESSAGES (EXPORTING)	TYPE	BUSLOCALMSG_T (local partner id, local partner GUID, application name + Fields of structure BAPIRET2)	Message table

The unblocking check signature has 2 importing parameters namely, IT_PARTNERS_GUID & IV_APPL. Through the IT_PARTNERS_GUID parameter the unblocking report will pass the list of business partners to be checked either in the form of a partner id and a partner GUID. The reason to pass both is to give the consuming applications the flexibility to work with either of the unique identifiers whichever is suited for the applications. Through the IV_APPL parameter the relevant application name would be passed to the unblocking check module as optional information.

The end-of-purpose signature has 1 exporting parameters namely, ET_UNBLK_STATUS. Basically the unblocking report expects for each business partner where unblocking is allowed an entry with the provided business partner key information in the field PARTNER and the value 'X' in the field UNBLK_STATUS to allow unblocking.

4.4 ENTERPRISE SERVICE METHODS

To integrate external applications/systems (e.g. non ABAP systems) into the blocking and unblocking reports, the following web services are provided:

- Service called in blocking report:
 - While EoP check to collect EoP results from registered service providers.
 - While blocking to notify registered service providers about a successful blocking.
 - While blocking report in a depending system with interim mode to notify a registered service provider (leading system) with the interim EoP check results.
- Service called in unblocking report:
 - While unblock check to collect unblock permission from registered service providers.
 - While unblock to notify registered service providers about a successful unblock.

The blocking and unblocking functionality of the central business partner provides the following service methods in the namespace <http://sap.com/xi/ABA>:

Service Consumer	Service Provider	Purpose
------------------	------------------	---------

ABABusinessPartnerEOPRemoteOut	ABABusinessPartnerEOPRemoteIn	Integration for the EoP check with dependent systems
ABABusinessPartnerEOPInterimOut	ABABusinessPartnerEOPInterimIn	Notify master system about local EoP check interim results from a dependent system
ABABusinessPartnerEOPComplout	ABABusinessPartnerEOPComplIn	Notify dependent systems about blocked business partners together with corresponding SoRT information
ABABusinessPartnerEOPUnblkOut	ABABusinessPartnerEOPUnblkIn	Integration for the unblocking check with dependent systems
ABABusinessPartnerEOPResetOut	ABABusinessPartnerEOPResetIn	Notify dependent systems about unblocked business partners

The blocking and unblocking functionality of the central business partner uses the listed service consumer methods. If the corresponding service configuration is available, then all registered service providers as counterparts will be executed in the receiving systems. The above listed service providers can be used in ABAP based systems (starting with release NW 7.40) for the integration of the blocking and unblocking functionality between business partners stored in a master and dependent systems. Any other system can in addition create own service providers in their own namespace with the same interface as the service consumers. By doing this it would be also possible to implement the integration for the central business partner EoP check functionality with non ABAP systems. For the integration of external systems the following three general use cases have to be supported:

1. The blocking report started in the leading system to collect EoP information from connected depending systems and notify after successful blocking.
2. The so called interim mode of blocking report which is called in a depending system to notify the master system about the recent EoP information.
3. The unblocking report, which is used to unblock requested business partners. The unblock report can be started in the leading system. Two different services are called while the unblock run. One, to let the connected system verify whether the unblocking is allowed and the second after a successful unblock in the master system.

4.4.1 END OF PURPOSE CHECK FOR CENTRAL BUSINESS PARTNER

The enterprise service for EoP check is a synchronous query-response service for the central business partner.

The **query**-part sends out in table element `PartnerRemote` the business partner ID and GUID (`PARTNER` and `PARTNER_GUID`) together with the business partner type (`PARTNER_TYPE`) and mapping information about the correct business partner ID in a target system as set in a possible implementation of `BAdI_BUP_PARTNER_KEYMAP` (`TARGET_SYSTEM`, `RECEIVER_PARTNER`, `RECEIVER_PARTNER_GUID`).

The query-part contains also the following elements to determine the processing conditions:

Parameter	Description
CHECK_ALL_APP	'X' when despite of the occurrence of ongoing process also all other registered EoP checks have to be executed in the called system
TESTRUN	'X' when called in test mode

The **response**-part expects to receive on each level of data a result set of EoP results. The EoP results are to be returned in the table element `BusinessPartnerRemote`. The contained elements are the same as for the exporting parameter `ET_PUR_CMPLT_PARTNERS` of the “End-Of-Purpose” check interface as described before. Also the same kind of information is to be returned.

On the same level as the EoP check results is a log table element available (`BusinessPartnerRemoteMsg`) to return messages in case of complications.

4.4.2 BLOCK NOTIFICATION FOR CENTRAL BUSINESS PARTNER

At the end of the blocking run, a notification is sent out to all registered service providers of all blocked data. The message contains the table element `Partnercomplete`, which contains all collected SoRT information for the new blocked business partners. The contained elements are the similar as for the exporting parameter `ET_PUR_CMPLT_PARTNERS` of the “End-Of-Purpose” check interface as described before.

4.4.3 INTERIM CHECK RESULTS NOTIFICATION FOR CENTRAL BUSINESS PARTNER

Each connected (registered) dependent system is allowed to provide SoRT information proactive to the master system. These interims results have to be provided in the same structure as the response of the EoP check with the exception that no log information is possible. Concrete these means that the message contains the table element `Partnercomplete`, which contains all collected SoRT information. The contained elements are the similar as for the exporting parameter `ET_PUR_CMPLT_PARTNERS` of the “End-Of-Purpose” check interface as described before.

The master system takes over the role of service provider.

4.4.4 UNBLOCK CHECK FOR CENTRAL BUSINESS PARTNER

The unblock check service is query-response service. The service is called for the central business partner to let dependent systems confirm that the unblocking of the given business partner is allowed.

The **query**-part sends out in the table element `IT_PARTNERS` the business partner ID and GUID (`PARTNER` and `PARTNER_GUID`) together with the business partner type (`PARTNER_TYPE`) and mapping information about the correct business partner ID in a target system as set in a possible implementation of `BAdI BUP_PARTNER_KEYMAP (TARGET_SYSTEM, RECEIVER_PARTNER, RECEIVER_PARTNER_GUID)`.

The **response**-part expects to receive for each provided business partner in the table element `ET_UNBLK_STATUS` in the element `UNBLK_STATUS` the value 'X', when the permission to unblock is granted.

On the same level as the unblocking check results is a Log table element available (`ET_MESSAGES`) to return messages in case of complications.

4.4.5 UNBLOCK NOTIFICATION FOR CENTRAL BUSINESS PARTNER

At the end of the unblocking run, a notification is sent out to all registered service providers of all unblocked data. The message contains the table element `PartnerReset`, which contains the information for the unblocked business partners in the contained elements `PARTNER` and `PARTNER_GUID`.

4.5 BUSINESS ADD-INS (BADIS)

4.5.1 BUSINESS PARTNER KEY MAPPING

The Business Add-In (BAdI) `BUP_PARTNER_KEYMAP` is used to find out the business partner ID and GUID of the target/receiving system. This BAdI is called before the RFC or web service call to registered dependent systems is called for the following purposes:

- EoP check before blocking
- Check before unblocking
- Unblocking notification

The BAdI includes the method `GET_PARTNER_ID`.

Parameter	Type	Associated Type	Description
<code>CT_KEYMAP</code>	Changing	<code>BUKMAP_T</code>	Table for Business Partner Key Mapping

The changing parameter `CT_KEYMAP` can be enhanced for each target system with an additional entry when the business partner GUID is not unique for business partners in the same landscape. The ID information about the business partner in the local system are provided in the components `PARTNER` and `PARTNER_GUID`, the ID information of the target system can be added in the components `RECEIVER_PARTNER` and `RECEIVER_PARTNER_GUID` together with the corresponding name of the target system in component `TARGET_SYSTEM`.

4.5.2 RESTRICTION OF BUSINESS PARTNER FROM EOP CHECK

The Business Add-In (BAdI) `BUP_PARTNER_EXCLIST` is used to restrict business partners from going through the EoP check. The restriction is carried out when the blocking report removes the partners from the list that are included in the check. The BAdI is used as an optimization measure.

The BAdI includes the method `PARTNER_EXCLIST`.

Parameter	Type	Associated Type	Description
<code>IV_FLT_VAL</code>	Importing	<code>BUS_DA_SELECT_VARIANT</code>	Variant / BAdI Filter for Selection of Data

IV_DIFFERENTIATION	Importing	CHAR20	
CT_PARTNER_EXCLIST	Changing	BU_PARTNER_T	Table of Business Partner Numbers

The importing parameters `IV_FLT_VAL` and `IV_DIFFERENTIATION` provide filter information from the selection screen of the blocking report `BUPA_PREPARE_EOP` to implement the BAdI accordingly to these values and to exclude certain business partners. The changing parameter `CT_PARTNER_EXCLIST` contains the list of business partner IDs, which were selected according to the selection criteria information from the selection screen of the blocking report `BUPA_PREPARE_EOP`. Inside the BAdI implementation this list of business partners to be processed within the EoP check can be changed.

4.5.3 EXPORT SORT DETAILS OF BLOCKED/UNBLOCKED BP TO FS MEMORY

The Business Add-In (BAdI) `BUPA_PURPOSE_EXPORT` is used to export SoRT details of a blocked or unblocked business partner to the application-specific memory. The BAdI is called when the blocking indicator (field `XPCPT` in database table `BUT000`) is set or reset. The BAdI is not called for a partner when the business is ongoing.

The BAdI method `BUPA_PURCOMPL_EXPORT` is called in the blocking report and the RFC function module `BUPA_PURPOSE_COMPLETE` to replicate the unblocking information to registered dependent systems. A dependent application can use the method to set additional own blocking indicators, which should be set in addition to the blocking indicator of the business partner.

Parameter	Type	Associated Type	Description
IV_TIMESTAMP_BUS	Importing	TIMESTAMP	UTC time stamp
IV_CHECKMODE	Importing	BOOLE_D	
IT_BUTSORT	Importing	BU_BUTSORT_T	Table Type for BUTSORT
CT_RETURN	Changing	BUS_BAPIRET2_T	Business Partner: Table Type for BAPIRET2

The importing parameter `IV_TIMESTAMP_BUS` is set with the current UTC time stamp as information. The importing parameter `IV_CHECKMODE` is set with 'X', if the external data/memory handling in the application Financial Services is active. The parameter is only relevant for the delivered implementation of this application. The importing parameter `IT_BUTSORT` contains all SoRT information for the business partners, for which the blocking indicator is set. The changing parameter `CT_RETURN` provides an option to report errors, which occurred during the execution of the BAdI implementation.

The BAdI method `BUPA_PURERESET_EXPORT` is called in the unblocking report and the RFC function module `BUPA_PURPOSE_RESET` to replicate the unblocking information to registered dependent systems. A dependent application can use the method to reset additional own blocking indicators, which were set in addition to the blocking indicator of the business partner.

Parameter	Type	Associated Type	Description
IV_TIMESTAMP_BUS	Importing	TIMESTAMP	UTC time stamp
IV_CHECKMODE	Importing	BOOLE_D	
IT_PARTNERS	Importing	BU_PARTNER_T	Table of unblocked Business Partners
CT_RETURN	Changing	BUS_BAPIRET2_T	Business Partner: Table Type for BAPIRET2

The importing parameters IV_TIMESTAMP_BUS and IV_CHECKMODE and the changing parameter CT_RETURN have the same usage as for BAdI method BUPA_PURCOMPL_EXPORT. The importing parameter IT_PARTNERS contains the list of partners, for which the blocking indicator is reset.

4.6 ADAPTION BY DEPENDENT APPLICATIONS

A blocked business partner should not be changed or displayed by an un-authorized user. Compared to this can authorized users still display the details of a blocked business partner but should not be able to change the partner details. For this reason all internal change & read function modules of the central business partner which are consumed by the dependent applications are enhanced with a new optional importing parameter IV_REQ_BLK_MSG. The purpose of this new parameter is that the consuming applications must set "X" to this parameter while calling the central business partner function modules if they wish to get back an error message or an exception during an un-authorized access to the blocked business partner. If the parameter is not set and partner status is blocked then no message and no data would be returned.

The central business partner function modules would not throw the error message or an exception blindly because this could have disruptions in the consuming applications flow. The consuming applications are requested to adapt their calls to the central business partner modules in such a way that the necessary calls to the central business partner modules are called where necessary by setting this new optional parameter IV_REQ_BLK_MSG with "X" and handle the error message or exception which is coming out of the central business partner function modules.

Some examples of these function modules are:

- BUA_ADDRESS_DESCRIPTION_GET
- BUA_ADDRESS_GET
- BUA_ADDRESS_GET_ALL
- BUA_ADDRESS_READ
- BUA_ADDRESS_READ_ALL
- BUP_BANK_DESCRIPTION_GET
- BUP_BANK_GET
- BUP_BANK_GET_ALL
- BUP_CCARD_DESCRIPTION_GET
- BUP_CCARD_GET
- BUP_CCARD_GET_ALL
- BUP_PARTNER_DESCRIPTION_GET
- BUP_PARTNER_F4_INT_TABLE

- BUP_PARTNER_GET
- BUP_PARTNER_ROLES_GET

In difference to this are the changes to function modules, which are used for example within the BAPI methods of the following business objects:

- BUS1006 Business Partner
- BUS1006001 Business partner employee
- BUS1006002 Business Partner Contact Person Relationship
- BUS1006003 Business Partner Employee Relationship
- BUS1006004 Business Partner Group Hierarchy
- BUS1006006 Business Partner Shareholder Relationship

These function modules have also been authorization enabled to check for the unauthorized access to a blocked business partner. But the difference is that there is no new importing parameter `IV_REQ_BLK_MSG`. Instead for a blocked business partner an error message would be thrown through the "Return" table which refers to the structure `BAPIRET2`.

Some examples of these function modules are:

- BUPA_ADDRESSES_GET
- BUPA_ADDRESS_ADD
- BUPA_BANKDETAILS_GET
- BUPA_BANKDETAIL_ADD
- BUPA_CENTRAL_CHANGE
- BUPA_CENTRAL_GET_DETAIL
- BUPA_DESCRIPTION_GET
- BUPA_EXISTENCE_CHECK
- BUPA_IDENTIFICATIONDETAILS_GET
- BUPA_IDENTIFICATION_ADD
- BUPA_NUMBERS_GET
- BUPA_PARTNER_GET_BY_IDNUMBER
- BUPA_PCARD_ADD
- BUPA_PCARD_GETDETAIL
- BUPA_RELATIONSHIPS_GET
- BUPA_ROLES_GET
- BUPA_ROLE_EXISTENCE_CHECK_2
- BUPA_SEARCH

5 EOP CHECK FOR THE ERP CUSTOMER AND VENDOR

The following description should provide a guide for how to implement an EoP check specifically for the ERP customer and vendor. The general logic is described in the previous chapter "How-To of an EoP Check Implementation". This chapter concentrates on the ERP customer and vendor specific customizing, interfaces, BAdIs and supporting APIs.

5.1 REGISTRATION OF APPLICATIONS

Every consuming application of the ERP customer and vendor which wishes to ILM enable the business partner deletion process needs to first register its application name. The blocking process completely works based on the applications which have been registered.

For the registration start transaction CVP_CUST or start transaction SPRO and choose the following path:
Logistics - General → Business Partner → Deletion of Customer and Vendor Master Data
 Add the application name to the IMG entity “Define and Store Application Names for EoP Check”.
 Make sure to add an entry for each relevant ID type!

ID Type	Appl Name	Application Description
1 Customer master data	ERP_CUST	Customer Master
1 Customer master data	ERP_FI	Financial Accounting Documents
1 Customer master data	ERP_SD	Sales & Distribution
1 Customer master data	ZAB	Test
1 Customer master data	ZTEST	
2 Vendor master data	ERP_SD	Sales & Distribution
2 Vendor master data	ERP_VEND	Vendor Master
3 Contact Person	ERP_SD	Sales & Distribution

Figure 5-1 Example for the customizing of application names of the ERP customer and vendor

5.2 THE “END-OF-PURPOSE” CHECK INTERFACE

The ERP customer and vendor blocking functionality enables registered applications to execute different actions during the EoP check process respectively during the unblocking process of ERP customer or vendors. For this purpose the class interface CVP_IF_APPL_EOP_CHECK is delivered. Each application will have to develop a class implementing this interface and needs to register the class in the corresponding customizing.

For the registration in the customizing start transaction CVP_CUST or start transaction SPRO and choose the following path:

Logistics - General → Business Partner → Deletion of Customer and Vendor Master Data

Add the application class name to the IMG entity “Define Application Classes Registered for EoP Check”.

Make sure to add an entry for the previously defined application name and each relevant ID type! Mark the organizational level (General/Company code), for which the EoP check can be done in the application.

ID Type	Appl Name	Application Description	Registered class for EoP checks	General	Comp....
1 Customer master da...	ERP_CUST	Customer Master	CVP_CL_EOP_CHECK_MD_CUST	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1 Customer master da...	ERP_SD	Sales & Distribution	CL_CVP_SD_EOP_CHECK	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2 Vendor master data	ERP_SD	Sales & Distribution	CL_CVP_SD_EOP_CHECK	<input checked="" type="checkbox"/>	<input type="checkbox"/>
2 Vendor master data	ERP_VEND	Vendor Master	CVP_CL_EOP_CHECK_MD_VEND	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3 Contact Person	ERP_SD	Sales & Distribution	CL_CVP_SD_EOP_CHECK	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 5-1 Example for the customizing of application classes registered for EoP check of the ERP customer and vendor

5.2.1 INTERFACE METHODS

The Interface CVP_IF_APPL_EOP_CHECK defines the several instance methods. The instantiation of the implementation class is done at the start of the EoP check. The same instance will be used until the end of the process. In other words, applications can rely on their instance data (buffering...).

The interface methods relevant for the “End-Of-Purpose” Check process are:

Method name	Level	Description
INITIALIZE	Instance method	Initialize EoP Check, to get Execution Parameters
CHECK_PARTNERS	Instance method	EoP Check Partner
CHECK_CPD_PARTNERS	Instance method	EoP Check One Time Account (CPD) Partner
PARTNERS_PURCMPL_EXPORT	Instance method	Store Application Specific Blocking Information
FINALIZE	Instance method	Finalize EoP Check, for Cleanup and Dequeue Handling

5.2.2 INTERFACE ATTRIBUTES

The Interface CVP_IF_APPL_EOP_CHECK provides in addition several constant attributes in order avoid bugs according to wrong hardcoded values. Some examples for useful attributes are:

Attribute	Type	Description
GC_TYPE_CUSTOMER	CVP_ID_TYPE	Type customer
GC_TYPE_VENDOR	CVP_ID_TYPE	Type vendor
GC_TYPE_CONTACT	CVP_ID_TYPE	Type contact person
GC_CPD_TYPE_CUSTOMER	CVP_ID_TYPE_CUSTVEND	Type One time account (CPD): Customer
GC_CPD_TYPE_VENDOR	CVP_ID_TYPE_CUSTVEND	Type One time account (CPD): Vendor
GC_PUR_COMPL_NA	CVP_PUR_COMPL_STATUS	n/a (no Business made with partner at all)
GC_PUR_COMPL_NO	CVP_PUR_COMPL_STATUS	No (Business is ongoing with partner)
GC_PUR_COMPL_YES	CVP_PUR_COMPL_STATUS	Yes (Business is complete with partner/Residence time over)

5.2.3 INITIALIZE AND FINALIZE METHODS

INITIALIZE

Registered applications may need to perform some initialization and also need to know what the current parameters of the blocking report are. This method will be called directly after instantiation of the registered application class.

Parameter	Type	Associated Type	Description
IV_APPL	Importing	CVP_APPL_NAME	Application Name
IV_EOP_RUN_MODE	Importing	CVP_EOP_RUN_MODE	Execution Mode for EoP Check
IV_TEST_MODE	Importing	CVP_TEST_MODE	Test mode
IV_VALIDATE_NXTCHK_DATE	Importing	CVP_VALIDATE_NXTCHK_DATE	Validate Next Check Date Information during EoP Check
IV_CHECK_PUR_COMPL_STATUS	Importing	CVP_CHECK_PUR_COMPL_STATUS	Check for Purpose Completion Status
IV_PROT_OUTPUT	Importing	CVP_PROT_OUTPUT	EoP Check: Output of Object Logs
IV_DETAIL_LOG	Importing	CVP_DETAIL_LOG	EoP Check: Detail Log

FINALIZE

Registered application class may need to perform some cleanup and dequeue activities. This method will be called before destructing the class instance.

It provides as importing parameters the tables already used at call time of EoP methods CHECK_PARTNERS and CHECK_CPD_PARTNERS.

Parameter	Type	Associated Type	Description
IT_EOP_CHECK_PARTNERS	Importing	CVP_TT_EOP_CHECK_PARTNERS	Master data IDs for blocking

IT_EOP_CHECK_PARTNERS_CP D	Importing	CVP_TT_EOP_CHECK_PARTNERS_CP D	Master data IDs One Time account for blocking
-------------------------------	-----------	-----------------------------------	---

5.2.4 EOP CHECK METHOD CHECK_PARTNERS

This is the “End of Purpose” check method. Applications are informed about the partners requested to be checked for the end of purpose in IT_EOP_CHECK_PARTNERS. In ET_SORT_RESULT_PARTNERS applications have to return their check results. Occurred errors other further information messages are expected to be in ET_MESSAGES.

Parameter	Type	Associated Type	Description
IV_APPL	Importing	CVP_APPL_NAME	Application Name
IT_EOP_CHECK_PARTNERS	Importing	CVP_TT_EOP_CHECK_PARTNERS	Master data IDs for blocking
ET_SORT_RESULT_PARTNERS	Exporting	CVP_TT_SORT_RESULT	SORT information after EOP check
ET_MESSAGES	Exporting	CVP_TT_EOP_CHECK_MESSAGES	Result Log Messages after EOP check

The importing parameter IV_APPL will contain the registered application name as customized. This is needed for checks but also to enable (if possible) reuse of implementation for several different applications.

The importing parameter IT_EOP_CHECK_PARTNERS will contain a list of customer, vendor or contact person IDs, for which the EoP check has to be executed. The partner entries are differentiated by the value in field ID_TYPE (1 = Customer master data, 2 = Vendor master data, 3 = Contact Person). The field BUKRS contains the company code, for which a selected customer or vendor is maintained respectively is to be checked for the EoP of all application data related to this company code. The field VKORG is currently not used and will be always initial. The field ACCOUNT_GROUP contains the account group, for which the customer or vendor is created and is required as input value for the end of residence period calculation during the EoP check.

The provided entries in the importing parameter IT_EOP_CHECK_PARTNERS depend on the execution mode of the blocking report CVP_PREPARE_EOP. The report can be executed only for customers or vendors, but not for both at the same time. So the importing parameter will contain only entries for customer or vendors and the related contact persons. Further are the provided entries depending on the selection parameter “Data to be processed”. The report provides the following options to select the data to be processed:

1. *Process all data levels:*

The blocking report will select all relevant data for the selected customers or vendors on general level (corresponds to database table KNA1/LFA1, on company code level (KNB1/LFB1) and all existing contact persons (KNVK).

2. *Process only company code specific data:*
The blocking report will select all relevant data for the selected customers or vendors only on company code level (KNB1/LFB1).
3. *Process only contact persons:*
The blocking report will select all relevant data for the selected customers or vendors only all existing contact persons (KNVK).

In addition are the provided entries in the importing parameter `IT_EOP_CHECK_PARTNERS` depending on the registration of the EoP check class for the current application name in the customizing. This depends on the customized ID type and the supported organizational levels (general and or company code).

The end-of-purpose signature has 1 exporting parameters namely, `ET_PUR_CMPLT_PARTNERS`. Basically the blocking report expects the details which are necessary to be stored in the central table `CVP_SORT` for each entry in importing parameter `IT_EOP_CHECK_PARTNERS` to be checked, based on the “key” information fields `ID`, `ID_TYPE` and `BUKRS`. Please read through the following paragraph to get know more about the individual fields of the type:

- `BUSINESS_SYSTEM` is the system where the end-of-purpose check has been executed for residence rule calculation, this is important in the remote run of the blocking report where the same partner’s end-of-purpose needs to be calculated in a distributed environment. Usually to be returned is the logical name of the local system, which can be determined like in following example:

```
DATA: l_own_logical_system type logsys,
      call function 'OWN_LOGICAL_SYSTEM_GET'
            importing
              own_logical_system          = l_own_logical_system
            exceptions
              own_logical_system_not_defined = 1
              others                        = 2.
if sy-subrc <> 0.
  l_own_logical_system = sy-sysid.
endif.
```

- `APPLICATION_NAME` is also needed to know the partner consumption information for each application. The importing parameter `IV_APPL` provides the relevant application name, for which the registered function module was called.

Within an application if there are many sub-application for which the end-of-purpose needs to be calculated individually, the applications can make use of the “application rule variant” concept to get more granularities. For example, if there are application rule variants maintained for an application then the `SORT` information is needed for each rule variant of that application, that is relevant due to the application own customizing for a checked business partner.

- The purpose completion code should be returned in the field `PUR_CMPL_STATUS`. Application needs to fill the purpose completion code either with:
 - 1 = n/a (no business made with BP at all),
 - 2 = no (business is ongoing with BP),
 - 3 = yes (business is complete with BP and residence time is over).
- In the case of the residence date being met during the calculation by ILM, then the `ST_RET_DATE` in the form of a date needs to be filled and returned to the blocking report together with `PUR_CMPL_STATUS = 3` (business is complete with BP and residence time is over).
- `NEXTCHKDT` – “Next Check Date” is the concept which has been introduced for optimization reasons. To put more light into this concept, imagine there 1000 partners which are checked for end-of-purpose and only 500 partners have reached end-of-purpose successfully. The remaining 500 partners imply there are still open business transactions happening by the consuming applications. The applications which return a “Business Ongoing” status through the structure field `PUR_CMPL_STATUS` for these partners have to send a date which should be greater than the current date, which would be treated as the next check date for this partner for going through the process of end-of-purpose again.

If the applications responsible for returning status “Business Ongoing” does not set the “Next Check

Date" value, the central blocking report would default the next check date based on the central customizing. The defaulting by the central blocking report would be done based on the following logic:

In the blocking report once the purpose check is completed for each application, a check would happen to find whether the applications reporting ongoing business are returning a next check date. If the next check date is not returned, the report would read the value maintained in the table CVP_NXTCHK. In this customizing a value for a time period can be maintained, which will be used to determine the default next check date starting from the current date. The calculated next check date is then stored in the SoRT data returned by the corresponding application. If the application returns a check date, which is larger than the calculated default next check date due to the customizing, then the check date set by the application is overwritten with the calculated default next check date to avoid too long periods until the next end of purpose check for the corresponding business partner is done.

5.2.5 BLOCKING METHOD PARTNERS_PURCMPL_EXPORT

After the EoP check has run, the calling report may block some data. This method will provide to registered applications the list of data that will be blocked in parameter IT_EOP_PARTNER_PURCMPL_EXP. Application can use this to set further application specific blocking indicators. Occurring errors for example during the update in the application are expected to be returned in ET_MESSAGES.

Parameters	Type	Associated Type	Description
IT_EOP_PARTNER_PURCMPL_EXP	Importing	CVP_TT_EOP_PARTNER_PURCMPL_EXP	Master data IDs for blocking export interface
ET_MESSAGES	Exporting	CVP_TT_EOP_CHECK_MESSAGES	Result Log Messages after EOP check

5.2.6 BLOCKING PROCESS FLOW

The process flow below this provides an idea how the blocking report runs and at which time and for which purpose calls to registered applications are done.

Blocking Report starts

Loop for all registered applications

Method **INITIALIZE**

Provides info about the parameters of the blocking report. Applications can implement their own initialization

Endloop

Loop for all registered applications

Method **CHECK_PARTNERS**

Application to perform EoP checks for normal customer/vendors

Method **CHECK_CPD_PARTNERS**

Application to perform EoP checks for OneTime customer/vendors

Endloop
 Loop for all registered applications
 Method **PARTNERS_PURCMPL_EXPORT**
 Provides information to application about partners to be blocked
 Endloop
 Loop for all registered applications
 Method **FINALIZE**
 Application to release their own objects (enqueue, memory ...)
 Endloop
 Blocking report ends

5.3 THE UNBLOCKING CHECK INTERFACE

The ERP customer and vendor blocking supports also unblocking of partners. Application may do a consistency check before unblocking is allowed or need to also unblock some of their blocked data in related with unblocked partners. For this, several methods have been introduced in the CVP_IF_APPL_EOP_CHECK interface. The interface methods relevant for the unblocking process are:

Method name	Level	Description
INITIALIZE_UNBLOCK	Instance method	Initialize Unblocking, to get Execution Parameters
PARTNERS_PREP_PURCMPL_RESET	Instance method	Prepare unblock partners: Applications to confirm
PARTNERS_PURCMPL_RESET	Instance method	Unblock partners: Applications get informed
FINALIZE_UNBLOCK	Instance method	Finalize Unblocking, for Cleanup and Dequeue Handling

5.3.1 INITIALIZE AND FINALIZE METHODS FOR UNBLOCKING

INITIALIZE_UNBLOCK

Registered applications may need to perform some initialization and also need to know what the current parameters of the unblocking report are. This method will be called directly after instantiation of the registered application class.

Parameter	Type	Associated Type	Description
IV_APPL	Importing	CVP_APPL_NAME	Application Name
IV_TEST_MODE	Importing	CVP_TEST_MODE	Test mode

IV_PROCESS_DIRECTION	Importing	CVP_SPECIFIC_PROCESS_DIRECTION	Specific process direction
IV_PROT_OUTPUT	Importing	CVP_PROT_OUTPUT	EoP Check: Output of Object Logs
IV_DETAIL_LOG	Importing	CVP_DETAIL_LOG	EoP Check: Detail Log

FINALIZE_UNBLOCK

Registered application class may need to perform some cleanup and dequeue activities. This method will be called before destructing the class instance. It provides as importing parameter the table already used at call time of method `PARTNERS_PURCMPL_RESET`.

Parameter	Type	Associated Type	Description
IT_PARTNERS	Importing	CVP_TT_EOP_PARTNER_PURCMPL_EXP	Unblocked partners

5.3.2 UNBLOCKING METHODS

PARTNERS_PREP_PURCMPL_RESET

This is the preparation method. Applications are informed about the partners requested to be unblocked. In `ET_PARTNERS` applications may refuse to unblock a partner. The reason and explanation are expected to be in `ET_MESSAGES`.

Parameter	Type	Associated Type	Description
IT_PARTNERS	Importing	CVP_TT_EOP_PARTNER_PURCMPL_EXP	Partners to be unblocked
ET_PARTNERS	Exporting	CVP_TT_UNBLOCK_STATUS	Partners status for unblocking
ET_MESSAGES	Exporting	CVP_TT_EOP_CHECK_MESSAGES	Result Log Messages after unblocking preparation

PARTNERS_PURCMPL_RESET

This method provides information to applications about the partners to be unblocked in `IT_PARTNERS`. Occurring errors for example during the update in the application are expected to be returned in `ET_MESSAGES`.

Parameters	Type	Associated Type	Description
IT_PARTNERS	Importing	CVP_TT_EOP_PARTNER_PURCMPL_EXP	Partners to be unblocked
ET_MESSAGES	Exporting	CVP_TT_EOP_CHECK_MESSAGES	Result Log Messages after EOP check

5.3.3 UNBLOCKING PROCESS FLOW

The process flow below should provide an idea how the unblocking report will run and how it will call registered applications.

Unblocking report starts

Loop for all registered applications

Method INITIALIZE_UNBLOCKED: Provides info about the parameters of the Unblocking report. Applications can implement their own initialization

Endloop

Loop for all registered applications

Method PARTNERS_PREP_PURCMPL_RESET: Application checks if partners can be unblocked

Endloop

Loop for all registered applications

Method PARTNERS_PURCMPL_RESET: Provides information to application about partners to be unblocked

Endloop

Loop for all registered applications

Method FINALIZE_UNBLOCK: Application to release their own objects (enqueue, memory ...)

Endloop

Unblocking report ends

5.4 SPECIAL SCENARIO: ONE TIME ACCOUNTS

Special kinds of account groups are one-time accounts (CPD functionality). These are accounts that have a business transaction with you only once or rarely. One-time accounts require no master records, because you do not need this master record after the business transaction. You create collective respectively template master records for one-time customers and one-time vendors, where certain data (such as address and bank details) is not entered in the master record, but in the actual document.

Example

You create a collective master record for a dummy customer that only includes data for all customers in a certain region. This collective master record can include the following fields (e.g. Master record name, Language, Currency, Sales office processing the data).

If a one-time customer from this region orders goods from your company, use the customer number of the collective master record when processing the sales order. You enter the address and other data that is not in the collective master record in the sales order.

Consequences for the EoP check

The EoP check and the blocking for an application object created for a one-time-account is still triggered via the central blocking report. This has the advantage, that there is a central entry point to block these application objects (as the template master data can't be blocked). This makes sense also due to the fact that there is no reason to use any different residence rules as compared to a "normal" ERP customer or vendor.

The difference is that each application object has to block its own data at the end of their specific residence period. It is in this case not possible to indicate the blocking status via the master data. Instead each application has to introduce its own blocking indicator, which is set via the central blocking report. The application specific EoP check can set this flag directly (except of the active test mode). A differentiation between overall or interim check modes is not relevant, as always the data will be blocked only locally.

5.4.1 EOP CHECK METHOD CHECK_CPD_PARTNERS

This is the "End of Purpose" check method called for master records, which are marked for the usage as one-time-account. Applications are informed about the partners requested to be checked for the end of purpose in IT_EOP_CHECK_PARTNERS_CPD. Occurred errors and success messages are expected to be returned in ET_MESSAGES.

Parameters	Type	Associated Type	Description
IV_APPL	Importing	CVP_APPL_NAME	Application Name
IT_EOP_CHECK_PARTNERS_CPD	Importing	CVP_TT_EOP_CHECK_PARTNERS	Master data IDs for blocking
IV_TEST_MODE	Importing	CVP_TEST_MODE	Test Mode
ET_MESSAGES	Exporting	CVP_TT_EOP_CHECK_MESSAGES	Result Log Messages after EOP check

5.5 ENTERPRISE SERVICE METHODS

To integrate external applications/systems (e.g. non ABAP systems) into the blocking and unblocking reports, the following Enterprise Services (ES) are provided:

- Service called in blocking report:
 - While EoP check to collect EoP results from registered service providers.
 - While blocking to notify registered service providers about a successful blocking.
 - While blocking report in a depending system with interim mode to notify a registered service provider (leading system) with the interim EoP check results.
- Service called in unblocking report:
 - While unblock check to collect unblock permission from registered service providers.
 - While unblock to notify registered service providers about a successful unblock.

The blocking and unblocking functionality of the ERP customer provides the following service methods in the namespace `http://sap.com/xi/APPL/Global2`:

Service Consumer	Service Provider	Purpose
CustomerERPEndOfPurposeCheckQueryResponse_Out	CustomerERPEndOfPurposeCheckQueryResponse_In	Integration for the EoP check with dependent systems
CustomerERPEndOfPurposeInterimCheckNotification_Out	CustomerERPEndOfPurposeInterimCheckNotification_In	Notify master system about local EoP check interim results from a dependent system
CustomerERPEndOfPurposeBlockNotification_Out	CustomerERPEndOfPurposeBlockNotification_In	Notify dependent systems about blocked customers together with corresponding SoRT information
CustomerERPEndOfPurposeUnblockCheckQueryResponse_Out	CustomerERPEndOfPurposeUnblockCheckQueryResponse_In	Integration for the unblocking check with dependent systems
CustomerERPEndOfPurposeUnblockNotification_Out	CustomerERPEndOfPurposeUnblockNotification_In	Notify dependent systems about unblocked customers

The blocking and unblocking functionality of the ERP vendor (supplier) provides the following service methods in the namespace `http://sap.com/xi/APPL/Global2`:

Service Consumer	Service Provider	Purpose
SupplierERPEndOfPurposeCheckQueryResponse_Out	SupplierERPEndOfPurposeCheckQueryResponse_In	Integration for the EoP check with dependent systems
SupplierERPEndOfPurposeInterimCheckNotification_Out	SupplierERPEndOfPurposeInterimCheckNotification_In	Notify master system about local EoP check interim results from a dependent system
SupplierERPEndOfPurposeBlockNotification_Out	SupplierERPEndOfPurposeBlockNotification_In	Notify dependent systems about blocked suppliers together with corresponding SoRT information
SupplierERPEndOfPurposeUnblockCheckQueryResponse_Out	SupplierERPEndOfPurposeUnblockCheckQueryResponse_In	Integration for the unblocking check with dependent systems
SupplierERPEndOfPurposeUnblockNotification_Out	SupplierERPEndOfPurposeUnblockNotification_In	Notify dependent systems about unblocked suppliers

The blocking and unblocking functionality of the ERP customer and vendor uses the listed service consumer methods. If the corresponding service configuration is available, then all registered service providers as counterparts will be executed in the receiving systems. The above listed service providers can be used in ABAP based systems (starting with release ERP 6.0 EHP7 SP06) for the integration of the blocking and unblocking functionality between ERP customers and vendors stored in a master and dependent systems. Any other system can in addition create own service providers in their own namespace with the same interface as the service consumers. By doing this it would be also possible to implement the integration for the ERP customer and vendor EoP check functionality with non ABAP systems. For the integration of external systems the following three general use cases have to be supported:

1. The blocking report started in the ERP system as leading system to collect EoP information from connected depending systems and notify after successful blocking.
2. The so called interim mode of blocking report which is called in a depending system to notify the master system about the recent EoP information.
3. The unblocking report, which is used to unblock requested business partners. The unblock report can be started in the master system. Two different services are called while the unblock run. One, to let the connected system verify whether the unblocking is allowed and the second after a successful unblock in the master system.

5.5.1 END OF PURPOSE CHECK FOR CUSTOMER/SUPPLIER (VENDOR)

The enterprise service for EoP check is a synchronous query-response service supporting customer/supplier integrated into the business object ERP customer/ERP supplier.

The **query**-part sends out the customer/supplier (InternalID) on general level together with an indicator (CheckCustomerIndicator; CheckSupplierIndicator) whether indicates that a check on general level is requested or not. One level below the company code (ID) information is available as well as the relationship which contains the contact person (InternalID; ContactPersonInternalID) information that has to be checked.

The query-part contains a processing condition segment (ProcessingConditions). The ProcessingConditions parameters are in detail:

Parameter	Description
TestModeIndicator	'true' when called in test mode
FinalIndicator	'true' when called in final mode; 'false' indicates that the run is an interims check
SkipWhenOngoingBusinessIndicator	'true' when the first occurrence of ongoing process should lead to no further checks
ValidateNextCheckDateIndicator	'true' to skip further application check when date in future is already available or determined
CreateApplicationLogIndicator	'true' when creation of Application log is requested
LogDetailLevelCode	Values as listed as listID 11105 in Global Data Type Catalog (1 = No details; 3 = Abortions; 5 = Abortions and errors; 7 = Abortions, errors and warnings; 9 = all details)

The **response**-part expects to receive on each level of data a result set of EoP results. The EoP results are in detail:

Parameter	Description
ApplicationName	Name of the application returning the result information (mandatory)
RuleVariantName	Name of the ILM rule variant is used to calculate the dates
PurposeCompletionCode	Vales follows, needs to be inserted in the Global Data Type Catalog (1 = n/a (no Business at all); 2 = No (Business is ongoing); 3 = yes (Business is complete and residence time is over) (mandatory)
StartOfRetentionDate	Start of retention date
NextCheckDate	Data after which the next EoP check should be allowed

On the same level as the EoP check results is a log segment available to return messages in case of complications.

5.5.2 BLOCK NOTIFICATION FOR CUSTOMER/SUPPLIER

At the end of the blocking run, a notification is sent out to all registered service providers of all blocked data. On each level (customer/supplier, contact person, company code) an indicator is provided and is ,true', when the data is blocked.

5.5.3 INTERIM CHECK RESULTS NOTIFICATION FOR CUSTOMER/SUPPLIER

Each connected (registered) dependent system is allowed to provide SoRT information proactive to the master system. These interim results have to be provided in the same structure as the response of the EoP check with the exception that no log information is possible. Concrete these means on each level the ID of the given plus an EoP result segment.

The master system (ERP system) takes over the role of service provider.

5.5.4 UNBLOCK CHECK FOR CUSTOMER/SUPPLIER

The unblock check service is a query-response service. The service is called for ERP Customer / ERP Supplier to let dependent systems confirm that the unblocking of the given Customer/Vendor and belonging contact person is allowed.

The **query**-part sends out the customer/supplier (`InternalID`) on general level together with an indicator (`CheckCustomerIndicator`; `CheckSupplierIndicator`) whether indicates that a check on general level is requested or not. One level below the company code (`ID`) information is available as well as the relationship which contains the contact person (`ID`; `ContactPersonInternalID`) information needs to be checked.

This query-part contains a processing condition segment (`ProcessingConditions`) to influence general behavior.

The `ProcessingConditions` parameters are in detail:

Parameter	Description
<code>TestModeIndicator</code>	'true' when called in test mode
<code>CreateApplicationLogIndicator</code>	'true' when creation of Application log is requested
<code>LogDetailLevelCode</code>	Values as listed as listID 11105 in Global Data Type Catalog (<i>1 = No details; 3 = Abortions; 5 = Abortions and errors; 7 = Abortions, errors and warnings; 9 = all details</i>)

The **response**-part expects to receive on each level of data the indicator

`EndOfPurposeUnblockPermitIndicator` which is set to 'true' when permission to unblock is granted.

On the same level as the ID and the `EndOfPurposeUnblockPermitIndicator` a log segment is available to return messages in case of complications.

5.5.5 UNBLOCK NOTIFICATION FOR CUSTOMER/SUPPLIER

At the end of the unblocking run, a notification is sent out to all registered service providers of all unblocked data. On each level (customer/supplier, contact person, company code) an indicator

`EndOfPurposeUnblockIndicator` is provided and is 'true' when the data is blocked.

5.6 BUSINESS ADD-INS (BADIS)

5.6.1 MAPPING OF MASTER DATA ID FOR EOP AND UNBLOCKING

This Business Add-In (BAI) `CVP_EOP_MAP_REMOTE_PARTNERS` is used to map local ERP customer/vendor master data IDs from the master system from/to remote ERP customer/vendor master data IDs in the dependent systems.

The BAdI provides methods, which are called during the EoP preparation check or during master data unblocking. In addition there are in each case separate methods called from the master system or called from the dependent system as part of the RFC interface functionality. This allows mapping to the IDs of the connected system as specified always in the import parameter IV_MASTER_SYSTEM respectively IV_DEPENDENT_SYSTEM within this system, where the mapping information is available. The task of the implementation is to change before the execution the IDs in the changing parameters to the IDs, which are known in the called system. And to change it back afterwards to the original IDs of the calling system, so that the processing can continue there with the results of the called systems.

Methods called from the master system during EoP preparation

- EOP_PREP_OUTBOUND_MAP_BEFORE: This method permits to change at runtime the master data IDs (customer or vendor or contact persons) from the master system to the IDs of the dependent system.
- EOP_PREP_OUTBOUND_MAP_AFTER: This method permits to map in the master system the master data IDs and the data returned from the dependent system. The data that can be changed are the Start of Retention Time (SoRT) results per partners and the result log messages

Methods called from the dependent system via RFC during EoP preparation

- EOP_PREP_INBOUND_MAP_BEFORE: This method is triggered from the dependent system. It permits to map the master data IDs coming from the master system.
- EOP_PREP_INBOUND_MAP_AFTER: This method is triggered from the dependent system (called via RFC). It permits to map the SoRT results per master data before they are returned to the calling master system.

Methods called from the master system during unblocking of a customer, vendor or contact person

- UNBLOCK_OUTBOUND_MAP_BEFORE: This method permits to change at runtime the master data IDs (customer or vendor or contact persons) from the master system to the IDs of the dependent system.
- UNBLOCK_OUTBOUND_MAP_AFTER: This method permits to map in the master system the master data IDs and the data returned from the dependent system. The data that can be changed are the unblock partners results and the result log messages

Methods called from the dependent system via RFC during unblocking

- UNBLOCKING_INBOUND_MAP_BEFORE: This method is triggered from the dependent system. It permits to map the master data IDs coming from the master system.
- UNBLOCKING_INBOUND_MAP_AFTER: This method is triggered from the dependent system (called via RFC). It permits to map the unblocking results per master data before they are returned to the calling master system.

5.6.2 EOP CHECK REPORT - MODIFY SELECTION BEFORE AND AFTER DB SELECTION

The Business Add-In (BAdI) CVP_EOP_MODIFY_SELECTION is used to change the selection criteria of the initial screen of End of Purpose preparation report. It also permits to change the selected data of the report or completely replace the database selection of data. The BAdI is used as an optimization measure.

The BAdI includes the following two methods:

- BEFORE_SELECTION: This method permits to manipulate the data for selection purpose entered in the initial screen of the End of Purpose preparation report. It also permits to skip standard database selection.

Parameter	Type	Associated Type	Description
IV_FLAG_CUST_OR_VEND	Importing	CVP_ID_TYPE_CUSTVEND	Customer/vendor master data ID TYPE

IV_EOP_PROC_TYPE	Importing	CVP_EOP_PROC_TYPE	EoP Check: Data To Be Processed
IT_SO_KUNNR	Importing	TT_RANGE_KUNNR	Range of customer IDs
IT_SO_LIFNR	Importing	TT_RANGE_LIFNR	Range of Vendor IDs (LIFNR)
IT_SO_BUKRS	Importing	TT_RANGE_BUKRS	Range of Company Codes (BUKRS)
IT_SO_VKORG	Importing	TT_RANGE_VKORG	Range of Sales Organization (VKORG)
IT_SO_PARNR	Importing	TT_RANGE_PARNR	Range of Contact Persons IDs (PARNR)
EV_SKIP_DB_SELECTION	Exporting	BOOLE_D	Skip Database selection: TRUE (=X) and FALSE (=')
ET_EOP_CHECK_PARTNERS	Exporting	CVP_TT_EOP_CHECK_PARTNERS	Master data IDs for blocking
ET_EOP_CHECK_PARTNERS_CPD	Exporting	CVP_TT_EOP_CHECK_PARTNERS_CPD	Master data IDs One Time account for blocking
ET_MESSAGES	Exporting	CVP_TT_EOP_CHECK_MESSAGES	Result Log Messages after EOP check

- AFTER_SELECTION: This method permits to modify the data selected by the standard selection functionality or do the database selection using custom logic.

Parameter	Type	Associated Type	Description
IV_FLAG_CUST_OR_VEND	Importing	CVP_ID_TYPE_CUSTVEND	Customer/vendor master data ID TYPE
IV_EOP_PROC_TYPE	Importing	CVP_EOP_PROC_TYPE	EoP Check: Data To Be Processed
IT_SO_KUNNR	Importing	TT_RANGE_KUNNR	Range of customer IDs
IT_SO_LIFNR	Importing	TT_RANGE_LIFNR	Range of Vendor IDs (LIFNR)
IT_SO_BUKRS	Importing	TT_RANGE_BUKRS	Range of Company Codes (BUKRS)
IT_SO_VKORG	Importing	TT_RANGE_VKORG	Range of Sales Organization (VKORG)
IT_SO_PARNR	Importing	TT_RANGE_PARNR	Range of Contact Persons IDs (PARNR)
ET_EOP_CHECK_PARTNERS	Exporting	CVP_TT_EOP_CHECK_PARTNERS	Master data IDs for blocking

ET_EOP_CHECK_PARTNERS_CPD	Exporting	CVP_TT_EOP_CHECK_PARTNERS_CPD	Master data IDs One Time account for blocking
ET_MESSAGES	Exporting	CVP_TT_EOP_CHECK_MESSAGES	Result Log Messages after EOP check

5.6.3 CUSTOM CHANGE DOCUMENTS HANDLING FOR EOP CHECK

This Business Add-In (BAI) *CVP_CHG_DOC* is called during the execution of the end of purpose check for the registered application names *ERP_CUST* or *ERP_VEND*. During this process, *ERP_CUST* or *ERP_VEND* have to determine the last change date of a customer or vendor master data based on the stored change documents. The BAI permits to handle custom based change document.

There are 3 different methods:

- *TABLE_RELEVANCE*: This method permits to filter or change tables that have to be considered. It also permits to add custom tables via changing parameter *CT_CHDOC_TABLE_RELEVANCE*.
- *CHG_DOC_IS_RELEVANT*: This method permits to inform the process about the fact a given change document record is relevant to be considered. This can be returned via changing parameter *CV_IS_RELEVANT*.
- *OWN_CHANGE_DOC_DATES*: This method permits to change the determined creation date (*CV_CREATION_DATE*) and last change date (*CV_LAST_CHG_DATE*) of a given customer or vendor master data.

5.7 ADAPTION BY DEPENDENT APPLICATIONS

As blocked customer or vendor master data should not be changed respectively should not be displayed by an un-authorized user. To simplify the adaption by the dependent applications, all internal read function modules are enhanced. The default behavior of adapted read function modules is:

- return the block flag if available
- initialize personal and private data
- don't touch any technical field

Most of the adapted read function modules have a new import parameter *IV_BEHAVIOR* to set for a specific call one of the following behavior modes:

- '' = use global default behavior
- 1 = block flag is returned, personal and private data are initialized
- 2 = (only for single read API), raise a new exception when the data is blocked and the user is not authorized
- 3 = return data as if the data privacy enhancement was not active. (like before)
- 4 = (only for mass read API) filter blocked data

For some application, there will be the need to set a different default behavior without changing the call of each read function module. This means to set a global behavior or the need to have a levelled behavior for the application. In order to support levelled default and global behavior, static class *CVP_CL_READ_API_DEFLT_BEHAVIOR* is available in order to manipulate the default behavior as a stack. An example of such a levelled stack behavior could be:

- Start Application
Default behavior is 1
 - *CVP_CL_READ_API_DEFLT_BEHAVIOR*=>PUSH behavior 2
From now on, the default behavior is 2
 - ...

- Go deeper in the application
 - Default behavior is still 2
 - ...
 - V_KNA1_SINGLE_READ
 - ...
 - Go deeper
 - CVP_CL_READ_API_DEFLT_BEHAVIOR=>PUSH behavior 3
Default behavior is now 3
 - ...
 - V_KNA1_SINGLE_READ
 - ...
 - CVP_CL_READ_API_DEFLT_BEHAVIOR=>PULL
Default Behavior is back to 2
 - ...
 - V_KNA1_SINGLE_READ
 - ...
- CVP_CL_READ_API_DEFLT_BEHAVIOR=>PULL
As the behavior stack is initial, the system is back to default behavior 1
- End Application

Remark: The IV_BEHAVIOR as specified on a call of an adapted read function module has top priority in comparison of the default current behavior.

5.7.1 ADAPTATION OF READ FUNCTION MODULES

Importing parameters

A new optional importing parameter IV_BEHAVIOR type CVP_API_BEHAVIOR is provided.

Exporting/Changing/tables parameters

If the corresponding returned data relates to a table where a block flag is available, then also the returned data structures are extended with the block flag (e.g.: returned structure contains mainly KNA1 data, but CVP_XBLOCK was not available).

The block flag in exporting or changing parameter of read function modules can be set with the following different values:

- ‘ ’
Not blocked
- ‘X’
Blocked, data masked or initialized OR special behavior according to optional parameter CVP_API_BEHAVIOR.
- ‘A’
Blocked, but data unmasked because current used is allowed to display blocked data.

The Domain CVP_XBLOCK will still be untouched (only values ‘ ’ and ‘X’ allowed), because the database will only support these 2 values.

5.7.1.1 Specific for Single Read Function Modules

The behavior mode 4 to filter out table entries for blocked data is not supported for single read function modules.

Single read function modules have to support behavior mode 2 to raise a new exception when the data is blocked and the user is not authorized. For this purpose a new exception will be raised, if the behavior 2 is active. If the functionality has already exceptions, then a new exception like KUNNR_BLOCKED is added.

Some examples of these function modules are:

- V_KNA1_SINGLE_READ

- LFA1_READ_SINGLE
- LFB1_READ_SINGLE
- KNB1_READ_SINGLE
- KNVK_SINGLE_READ
- KNA1_SINGLE_READER
- KNB1_SINGLE_READ
- KNVV_SINGLE_READ
- LFM1_SINGLE_READ
- LFM2_SINGLE_READ

5.7.1.2 Specific for Mass Read Function Modules

Mass read function modules can't support behavior mode 2.

Some examples of these function modules are:

- V_KNA1_ARRAY_READ
- WY_KNVK_ARRAY_READ
- WY_LFA1_ARRAY_READ
- KNA1VV_GENERIC_READ_CUSTOMER
- KNVV_GENERIC_READER_CUSTOMER
- LFM1_GENERIC_READ_LIFNR
- LFM2_GENERIC_READ_LIFNR

5.7.2 ADAPTATION OF CHECK FUNCTION MODULES

Function modules which check the existence of master data generally raise exception errors and/or return error message. When the checked functionality deals with blocked data, either an error is returned or either a new exception is returned.

Importing parameters

A new optional importing parameter `IV_BEHAVIOR` type `CVP_API_BEHAVIOR` is provided. This is to allow behavior mode 3. Mode 1 and 2 has for this kind of function modules the same meaning, as for both modes an exception is raised. If the functionality has already exceptions, then a new exception like `KUNNR_BLOCKED` is added.

Remark: In case the function module does not have exceptions, then behavior mode 2 has an adapted behavior.

Some examples of these function modules are:

- PI_BP_CONTACT_EXIST_CHECK
- PI_BP_CUSTOMER_EXIST_CHECK
- BAPI_CUSTOMER_EXISTENCECHECK
- CUSTOMER_EXISTS
- MM_PARTNER_ROLE_CHECK
- BAPI_VENDOR_EXISTENCECHECK

5.7.3 ADAPTATION OF SEARCH FUNCTION MODULES

The default behavior for search function modules is to not return blocked data.

Importing parameters

A new optional importing parameter `IV_BEHAVIOR` type `CVP_API_BEHAVIOR` is provided. This is to allow behavior mode 3. Mode 1 and 2 has for this kind of function modules the same meaning.

Some examples of these function modules are:

- CUSTOMER_SEARCH
- VENDOR_SEARCH

5.7.4 ADAPTATION OF BAPI METHODS/FUNCTION MODULES

For function modules used in BAPI methods no interface changes are done.

The behavior mode depends on, from where the function module is called:

- If the function module is called from the same system (normal call or destination NONE), then it has to behave like a read function module as described above.
- If the function module is called from outside, the behavior can have been set beforehand. In this case, the behavior mode will be used as described above. If no behavior was set:
 - For a single read or an existence check function module an error is returned if the data is blocked. So only the behavior mode 2 is used.
 - For a search function module the blocked data is filtered by default. So only behavior mode 4 is supported.

Some examples of these function modules are:

- BAPI_CUSTOMER_EXISTENCECHECK
- BAPI_CUSTOMER_FIND
- BAPI_CUSTOMER_GETDETAIL2
- BAPI_CUSTOMER_GETLIST
- BAPI_CUSTOMER_GETCONTACTLIST
- BAPI_BUSPARTNEREMPLOYEE_GETLIST
- BAPI_VENDOR_EXISTENCECHECK
- BAPI_VENDOR_FIND
- BAPI_VENDOR_GETDETAIL
- BAPI_CUSTOMER_GETDETAIL
- BAPI_CUSTOMER_GETDETAIL1
- BAPI_CUSTOMER_SEARCH

5.7.5 ADAPTATION OF ENTERPRISE SERVICES

For the adaption of enterprise services no interface changes are done. The following behavior has been implemented:

- Read services raise an error for blocked data
- Find services will filter results
- Other services have to consider blocked data. E.g. an update/change data service must not be able to change blocked data.

Some examples of enterprise services are:

WebService	Service Interface
ECC_CUSTOMERIDQR	CustomerERPByIDQueryResponse_In
ECC_CUSTBASICDATABYIDQR_V1	CustomerERPBasicDataByIDQueryResponse_In_V1
ECC_CUSTOMERBANKDETAILSIDDQR	CustomerERPBankDetailsByIDQueryResponse_In
ECC_CUSTOMCONTACTPERSONQR	CustomerERPRelationshipContactPersonByIDAndContactPersonInternalIDQueryResponse_In

ECC_CUSTBASICDATABYIDQR_V2	CustomerERPBasicDataByIDQueryResponse_In_V2
ECC_CUSTOMERERPSIMPLEBYIDQR	CustomerERPSimpleByIDQueryResponse_In
ECC_CUSTERPUNLDPT001QR	CustomerERPUnloadingPointNameByIDQueryResponse_In
ECC_CUSTERPRCVGPT001QR	CustomerERPReceivingPointPartyByIDQueryResponse_In
ECC_SALESARRANGEMENTIEQR	SalesArrangementERPByIdentifyingElementsQueryResponse_In
ECC_SALESARRANGEMENTSBCQR	SalesArrangementSimpleByCustomerIDQueryResponse_In
ECC_SUPPLIERSNAQR	SupplierSimpleByNameAndAddressQueryResponse_In
ECC_CUSTOMERERPSE001QR	CustomerERPBasicDataUpdateRequestConfirmation_In

6 EOP CHECK FOR THE SCM LOCATION

The following description should provide a guide for how to implement an EoP check specifically for the SCM location. The general logic is described in the previous chapter “How-To of an EoP Check Implementation”. This chapter concentrates on the SCM location specific customizing, interfaces and supporting APIs.

6.1 REGISTRATION OF APPLICATIONS

Every consuming application of the SCM location which wishes to ILM enable the business partner deletion process needs to first register its application name. The blocking process completely works based on the applications which have been registered.

For the registration start transaction /SCMB/LOC_CUST or start transaction SPRO in SCM and choose the following path:

SCM Basis → Master Data → Location → Location Master Data Deletion

Add the application name to the IMG entity “Register Application Names for EoP Check”.

Make sure to add an entry for each relevant location type!

Location Type	Application Name	Applic. Component	Application Description
1010 Customer	SCMB_MD_CHD	SCM-BAS-MD-LO	MD - Last Changed Date
1010 Customer	Z_TEPLATE_APPL	SCM-BAS-MD-LO	Template application
1011 Vendor	SCMB_MD_CHD	SCM-BAS-MD-LO	MD - Last Changed Date
1011 Vendor	Z_TEPLATE_APPL	SCM-BAS-MD-LO	Template application
1020 Transportation Service Provider	SCMB_MD_CHD	SCM-BAS-MD-LO	MD - Last Changed Date
1050 Subcontractor	SCMB_MD_CHD	SCM-BAS-MD-LO	MD - Last Changed Date
1010 Customer			
1011 Vendor			
1020 Transportation Service Provider			
1050 Subcontractor			

Figure 6-1 Example for the customizing of application names of the SCM location

6.2 THE “END-OF-PURPOSE” CHECK INTERFACE

The SCM location blocking functionality enables registered applications to execute different actions during the EoP check process respectively during the unblocking process of SCM location. For this purpose the class interface /SCMB/LOC_IF_APPL_EOP_CHECK is delivered. Each application will have to develop a class implementing this interface and needs to register the class in the corresponding customizing.

For the registration in the customizing start transaction /SCMB/LOC_CUST or start transaction SPRO and choose the following path:

SCM Basis → Master Data → Location → Location Master Data Deletion

Add the application class name to the IMG entity “Register Application Classes for EoP Check”.

Make sure to add an entry for the previously defined application name and each relevant location type! Mark the organizational level (General/Company code), for which the EoP check can be done in the application.

Change View "Register Application classes": Overview

Location Type	Application Name	Item Number	Application Description	Registered class for EoP checks	General	Comp. Code
1010 Customer	SCMB_MD_CHD	0	MD - Last Changed Date	/SCMB/LOC_CL_EOP_CHECK_LASTCHD	<input checked="" type="checkbox"/>	<input type="checkbox"/>
1010 Customer	Z_TEPLATE_APPL	0	Template application	Z_SCMB_LOC_TEMPL_IMPL_EOPCHECK	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1011 Vendor	SCMB_MD_CHD	0	MD - Last Changed Date	/SCMB/LOC_CL_EOP_CHECK_LASTCHD	<input checked="" type="checkbox"/>	<input type="checkbox"/>
1011 Vendor	Z_TEPLATE_APPL	0	Template application	Z_SCMB_LOC_TEMPL_IMPL_EOPCHECK	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1020 Transport...	SCMB_MD_CHD	0	MD - Last Changed Date	/SCMB/LOC_CL_EOP_CHECK_LASTCHD	<input checked="" type="checkbox"/>	<input type="checkbox"/>
1050 Subcontract...	SCMB_MD_CHD	0	MD - Last Changed Date	/SCMB/LOC_CL_EOP_CHECK_LASTCHD	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 6-2 Example for the customizing of application classes registered for EoP check of the SCM location

6.2.1 INTERFACE METHODS

The Interface /SCMB/LOC_IF_APPL_EOP_CHECK defines the several instance methods. The instantiation of the implementation class is done at the start of the EoP check. The same instance will be used until the end of the process. In other words, applications can rely on their instance data (buffering...).

The interface methods relevant for the “End-Of-Purpose” Check process are:

Method name	Level	Description
INITIALIZE	Instance method	Initialize EoP Check, to get Execution Parameters
CHECK_PARTNERS	Instance method	EoP Check Partner
CHECK_CPD_PARTNERS	Instance method	EoP Check One Time Account (CPD) Partner
PARTNERS_PURCMPL_EXPORT	Instance method	Store Application Specific Blocking Information
FINALIZE	Instance method	Finalize EoP Check, for Cleanup and Dequeue Handling

6.2.2 INTERFACE ATTRIBUTES

The interface /SCMB/LOC_IF_APPL_EOP_CHECK provides in addition several constant attributes in order to avoid bugs according to wrong hardcoded values. Some examples of useful attributes are:

Attribute	Type	Description
GC_TYPE_CUSTOMER	/SCMB/EOP_LOC_TYPE	Type customer
GC_TYPE_VENDOR	/SCMB/EOP_LOC_TYPE	Type vendor
GC_TYPE_TSP	/SCMB/EOP_LOC_TYPE	Type TSP
GC_TYPE_SUBCONTRACTOR	/SCMB/EOP_LOC_TYPE	Type Subcontractor
GC_PUR_COMPL_NA	/SCMB/LOC_PUR_COMPL_STATUS	n/a (no Business made with partner at all)
GC_PUR_COMPL_NO	/SCMB/LOC_PUR_COMPL_STATUS	No (Business is ongoing with partner)
GC_PUR_COMPL_YES	/SCMB/LOC_PUR_COMPL_STATUS	Yes (Business is complete with partner/Residence time over)

6.2.3 INITIALIZE AND FINALIZE METHODS

INITIALIZE

Registered applications may need to perform some initialization and also need to know what the current parameters of the blocking report are. This method will be called directly after instantiation of the registered application class.

Parameter	Type	Associated Type	Description
IV_APPL	Importing	/SCMB/LOC_APPL_NAME	Application Name
IV_TEST_MODE	Importing	/SCMB/LOC_TEST_MODE	Test mode
IV_VALIDATE_NXTCHK_DATE	Importing	/SCMB/LOC_VALIDATE_NXTCHK_DATE	Validate Next Check Date Information during EoP Check
IV_CHECK_PUR_COMPL_STATUS	Importing	/SCMB/LOC_CHECK_PUR_COMPL_STAT	Check for Purpose Completion Status
IV_PROT_OUTPUT	Importing	/SCMB/LOC_PROT_OUTPUT	EoP Check: Output of Object Logs
IV_DETAIL_LOG	Importing	/SCMB/LOC_DETAIL_LOG	EoP Check: Detail Log

FINALIZE

Registered application class may need to perform some cleanup and dequeue activities. This method will be called before destructing the class instance.

It provides as importing parameters the table already used at call time of EoP method CHECK_PARTNERS.

Parameter	Type	Associated Type	Description
IT_EOP_CHECK_PARTNERS	Importing	/SCMB/LOC_TT_EOPCHECK_PARTNERS	Master data IDs for blocking

6.2.4 EOP CHECK METHOD CHECK_PARTNERS

This is the “End of Purpose” check method. Applications are informed about the partners (in table IT_EOP_CHECK_PARTNERS) coming from business system group IV_LOGQS which are requested to be checked for the end of purpose. In ET_SORT_RESULT_PARTNERS applications have to return their check results. Occurred errors other further information messages are expected to be in ET_MESSAGES.

Parameter	Type	Associated Type	Description
IV_APPL	Importing	/SCMB/LOC_APPL_NAME	Application Name
IV_LOGQS	Importing	/SAPAPO/LOC_LOGQS	Location: Business System Group
IT_EOP_CHECK_PARTNERS	Importing	/SCMB/LOC_TT_EOPCHECK_PARTNERS	Location IDs for blocking
ET_SORT_RESULT_PARTNERS	Exporting	/SCMB/LOC_TT_SORT_RESULT	SORT information after EOP check
ET_MESSAGES	Exporting	/SCMB/LOC_TT_EOPCHECK_MESSAGES	Result Log Messages after EOP check

The importing parameter IV_APPL will contain the registered application name as customized. This is needed for checks but also to enable (if possible) reuse of implementation for several different applications.

The importing parameter IV_LOGQS specifies business system group to which locations in IT_EOP_CHECK_PARTNERS belongs.

The importing parameter IT_EOP_CHECK_PARTNERS will contain a list of SCM location IDs, for which the EoP check has to be executed. The partner entries are differentiated by the value in field ID_TYPE (1010 = Customer, 1011 = Vendor, 1020 = Transportation Service Provider, 1050 = Subcontractor). The field BUKRS contains the company code and is filled just for location created via integration from ERP system. If it is filled, the method should consider if there are objects with open business for the given location and any plant belonging to the company code.

The provided entries in the importing parameter IT_EOP_CHECK_PARTNERS depend on the execution mode of the blocking report /SCMB/RLOC_PREPARE_EOP. The report can be executed only for one location type at

a time. Further are the provided entries depending on the selection parameter *“Data to be processed”*. The report provides the following options to select the data to be processed:

1. *Process all data levels:*
The blocking report will select all relevant data for the selected locations on general level. Open business will be checked irrespective of plants involved in the business.
2. *Process only company code specific data:*
This mode is possible only for location integrated from ERP. The blocking report will select all relevant data for the selected integrated location and business system group. Open business will be checked for selected locations and plants belonging to selected company code(s).

In addition are the provided entries in the importing parameter `IT_EOP_CHECK_PARTNERS` depending on the registration of the EoP check class for the current application name in the customizing. This depends on the customized location type and the supported organizational levels (general and/or company code).

The end-of-purpose signature has exporting parameter `ET_SORT_RESULT_PARTNERS`. Basically the blocking report expects the details which are necessary to be stored in the central table `/SCMB/LOC_SORT` for each entry in importing parameter `IT_EOP_CHECK_PARTNERS` to be checked, based on the “key” information fields `ID`, `ID_TYPE` and `BUKRS`. Please read through the following paragraph to get know more about the individual fields of the type:

- `APPL_NAME` is needed to know the partner consumption information for each application. The importing parameter `IV_APPL` provides the relevant application name, for which the registered function module was called.
Within an application if there are many sub-application for which the end-of-purpose needs to be calculated individually, the applications can make use of the “application rule variant” concept to get more granularities. For example, if there are application rule variants maintained for an application then the `SORT` information is needed for each rule variant of that application, that is relevant due to the application own customizing for a checked business partner.
- The purpose completion code should be returned in the field `PUR_CMPL_STATUS`. Application needs to fill the purpose completion code either with:
 - 1 = *n/a (no business made with BP at all)*,
 - 2 = *no (business is ongoing with BP)*,
 - 3 = *yes (business is complete with BP and residence time is over)*.
- In the case of the residence date being met during the calculation by ILM, then the `ST_RET_DATE` in the form of a date needs to be filled and returned to the blocking report together with `PUR_CMPL_STATUS = 3` (business is complete with BP and residence time is over).
- `NEXTCHKDT` – “Next Check Date” is the concept which has been introduced for optimization reasons. To put more light into this concept, imagine there 1000 partners which are checked for end-of-purpose and only 500 partners have reached end-of-purpose successfully. The remaining 500 partners imply there are still open business transactions happening by the consuming applications. The applications which return a “Business Ongoing” status through the structure field `PUR_CMPL_STATUS` for these partners have to send a date which should be greater than the current date, which would be treated as the next check date for this partner for going through the process of end-of-purpose again.

If the applications responsible for returning status “Business Ongoing” does not set the “Next Check Date” value, the central blocking report would default the next check date based on the central customizing. The defaulting by the central blocking report would be done based on the following logic:

In the blocking report once the purpose check is completed for each application, a check would happen to find whether the applications reporting ongoing business are returning a next check date. If the next check date is not returned, the report would read the value maintained in the table `/SCMB/LOC_NXTCHK`. In this customizing a value for a time period can be maintained, which will be used to determine the default next check date starting from the current date. The calculated next check date is then stored in the `SoRT` data returned by the corresponding application. If the application returns a check date, which is larger than the calculated default next check date due to

the customizing, then the check date set by the application is overwritten with the calculated default next check date to avoid too long periods until the next end of purpose check for the corresponding business partner is done.

6.2.5 BLOCKING METHOD PARTNERS_PURCMPL_EXPORT

After the EoP check has run, the calling report may block some data. This method will provide to registered applications the list of data that will be blocked in parameter `IT_EOP_PARTNER_PURCMPL_EXP`. Application can use this to set further application specific blocking indicators. Occurring errors for example during the update in the application are expected to be returned in `ET_MESSAGES`.

Parameters	Type	Associated Type	Description
<code>IT_EOP_PARTNER_PURCMPL_EXP</code>	Importing	<code>/SCMB/LOC_TT_EOP_P_PURCMPL_EXP</code>	Location IDs for blocking export interface
<code>ET_MESSAGES</code>	Exporting	<code>/SCMB/LOC_TT_EOPCHECK_MESSAGES</code>	Result Log Messages after EOP check

6.2.6 BLOCKING PROCESS FLOW

The process flow below this provides an idea how the blocking report runs and at which time and for which purpose calls to registered applications are done.

Blocking Report starts

Loop for all registered applications

Method **INITIALIZE**

Provides info about the parameters of the blocking report. Applications can implement their own initialization

Endloop

Loop for all registered applications

Method **CHECK_PARTNERS**

Application to perform EoP checks for SCM location

Endloop

Loop for all registered applications

Method **PARTNERS_PURCMPL_EXPORT**

Provides information to application about SCM locations to be blocked

Endloop

Loop for all registered applications

Method **FINALIZE**

Application to release their own objects (enqueue, memory ...)

Endloop

Blocking report ends

6.3 THE UNBLOCKING CHECK INTERFACE

The SCM location blocking supports also unblocking of partners. Application may do a consistency check before unblocking is allowed if need to also unblock some of their blocked data in related with unblocked partners. For this, several methods have been introduced in the /SCMB/LOC_IF_APPL_EOP_CHECK interface. The interface methods relevant for the unblocking process are:

Method name	Level	Description
INITIALIZE_UNBLOCK	Instance method	Initialize Unblocking, to get Execution Parameters
PARTNERS_PREP_PURCMPL_RESET	Instance method	Prepare unblock partners: Applications to confirm
PARTNERS_PURCMPL_RESET	Instance method	Unblock partners: Applications get informed
FINALIZE_UNBLOCK	Instance method	Finalize Unblocking, for Cleanup and Dequeue Handling

6.3.1 INITIALIZE AND FINALIZE METHODS FOR UNBLOCKING

INITIALIZE_UNBLOCK

Registered applications may need to perform some initialization and also need to know what the current parameters of the unblocking report are. This method will be called directly after instantiation of the registered application class.

Parameter	Type	Associated Type	Description
IV_APPL	Importing	/SCMB/LOC_APPL_NAME	Application Name
IV_TEST_MODE	Importing	/SCMB/LOC_TEST_MODE	Test mode
IV_PROT_OUTPUT	Importing	/SCMB/LOC_PROT_OUTPUT	EoP Check: Output of Object Logs
IV_DETAIL_LOG	Importing	/SCMB/LOC_DETAIL_LOG	EoP Check: Detail Log

FINALIZE_UNBLOCK

Registered application class may need to perform some cleanup and dequeue activities. This method will be called before destructing the class instance. It provides as importing parameter the table already used at call time of method PARTNERS_PURCMPL_RESET.

Parameter	Type	Associated Type	Description
IT_PARTNERS	Importing	/SCMB/LOC_TT_EOP_P_PURCMPL_EXP	Location IDs for blocking export interface

6.3.2 UNBLOCKING METHODS

PARTNERS_PREP_PURCMPL_RESET

This is the preparation method. Applications are informed about the partners requested to be unblocked. In ET_PARTNERS applications may refuse to unblock a partner. The reason and explanation are expected to be in ET_MESSAGES.

Parameter	Type	Associated Type	Description
IV_LOGQS	Importing	/SAPAPO/LOC_LOGQS	Location: Business System Group
IT_PARTNERS	Importing	/SCMB/LOC_TT_EOP_P_PURCMPL_EXP	Partners to be unblocked
ET_PARTNERS	Exporting	/SCMB/LOC_TT_UNBLOCK_STATUS	Partners status for unblocking
ET_MESSAGES	Exporting	/SCMB/LOC_TT_EOPCHECK_MESSAGES	Result Log Messages after unblocking preparation

PARTNERS_PURCMPL_RESET

This method provides information to applications about the partners to be unblocked in IT_PARTNERS. Occurring errors for example during the update in the application are expected to be returned in ET_MESSAGES.

Parameters	Type	Associated Type	Description
IV_LOGQS	Importing	/SAPAPO/LOC_LOGQS	Location: Business System Group
IT_PARTNERS	Importing	/SCMB/LOC_TT_EOP_P_PURCMPL_EXP	Partners to be unblocked
ET_MESSAGES	Exporting	/SCMB/LOC_TT_EOPCHECK_MESSAGES	Result Log Messages after EOP check

6.3.3 UNBLOCKING PROCESS FLOW

The process flow below should provide an idea how the unblocking report will run and how it will call registered applications.

Unblocking report starts

Loop for all registered applications

Method INITIALIZE_UNBLOCKED: Provides info about the parameters of the Unblocking report. Applications can implement their own initialization

Endloop

Loop for all registered applications

Method PARTNERS_PREP_PURPCMPL_RESET: Application checks if partners can be unblocked

Endloop

Loop for all registered applications

Method PARTNERS_PURCMPL_RESET: Provides information to application about partners to be unblocked

Endloop

Loop for all registered applications

Method FINALIZE_UNBLOCK: Application to release their own objects (enqueue, memory ...)

Endloop

Unblocking report ends

6.4 ADAPTION BY DEPENDENT APPLICATIONS

Blocked SCM location master data should not be changed respectively should not be displayed by an unauthorized user. To simplify the adaption by the dependent applications, all internal location read function modules are enhanced. The default behavior of adapted read function modules is:

- initialize personal data
- return blocked locations with blocking status (LOC_XBLCK) in new exporting table ET_LOC_BLOCK_STATUS where:
 - 'X' is set for generally blocked location for which Authority check fails
 - 'A' is set for blocked location (general or company code level) for which authority check is successful
 - 'B' is set for a location for which a Company level Block exists. 'B' will only be set in a company code relevant case
 - Location without block flag will not be listed in table ET_LOC_BLOCK_STATUS

Adapted location read function modules can be controlled by static class

/SCMB/LOC_CL_READ_API_DEFLT_BV to behave in compliance with one of the following behavior modes:

- '' = use global default behavior
- 1 = block flag is returned, personal and private data are initialized
- 2 = (only for single read API), raise a new exception when the data is blocked and the user is not authorized
- 3 = return data as if the data privacy enhancement was not active. (like before)
- 4 = (only for mass read API) filter blocked data
- 5 = (only for mass read API) filter records referring to blocked location disregard the authorization

Static class /SCMB/LOC_CL_READ_API_DEFLT_BV can be used to manipulate the default behavior as a stack. An example of such a levelled stack behavior could be:

- Start Application
 - Default behavior is 1
 - /SCMB/LOC_CL_READ_API_DEFLT_BV=>PUSH behavior 2
From now on, the default behavior is 2
 - ...
Go deeper in the application
 - Default behavior is still 2
 - ...
 - Call of location read function module
 - ...
 - Go deeper
 - /SCMB/LOC_CL_READ_API_DEFLT_BV=>PUSH behavior 3
Default behavior is now 3
 - ...
 - Call of location read function module
 - ...
 - /SCMB/LOC_CL_READ_API_DEFLT_BV=>PULL
Default Behavior is back to 2
 - ...
 - Call of location read function module
 - ...
 - /SCMB/LOC_CL_READ_API_DEFLT_BV=>PULL
As the behavior stack is initial, the system is back to default behavior 1
- End Application

7 EOP RELATED CUSTOMIZING SETTINGS

7.1 AVAILABLE CUSTOMIZING SETTINGS (IMG AND ILM)

In the IMG (SPRO) the following settings are to be done:

Delivery by the Business Partner Dependent Application

- registered Application Name for the EoP check
- registered Function Module (cBP) or Application class for the EoP check (ERP Customer/Vendor, SCM Location)

To be maintained by SAP customers

- required Application Rule Variants for the EoP check
- RFC connection of the master system
- RFC connection of dependent systems
- default Next Check Date settings
- further cBP or ERP Customer/Vendor or SCM Location specific settings
- settings for unblocking check functionality

7.2 IMG SETTINGS - CENTRAL BUSINESS PARTNER DATA (CBP)

The settings for blocking and deletion of business partner master data are maintained in the customizing for *Cross-Application Components* under *Data Protection -> Blocking and Unblocking -> Business Partner*.

7.3 IMG SETTINGS – ERP CUSTOMER AND VENDOR

The settings related to the blocking and deletion of customer and vendor master data are maintained in the customizing via transaction `CVP_CUST` or in the IMG (SPRO) using one of the following paths:

- *Logistics - General -> Business Partner -> Deletion of Customer and Vendor Master Data.*
- *Financial Accounting -> Accounts Receivable and Accounts Payable -> Deletion of Customer and Vendor Master Data.*
- *Financial Accounting (New) -> Accounts Receivable and Accounts Payable -> Deletion of Customer and Vendor Master Data.*

7.4 IMG SETTINGS – SCM LOCATION

The settings related to the blocking and deletion of customer and vendor master data are maintained in the customizing via transaction `/SCMB/LOC_CUST` or in the IMG (SPRO) using the following path:

- *SCM Basis → Master Data → Location → Location Master Data Deletion.*

7.5 ILM SETTINGS

7.5.1 IRM_CUST (DELIVERED BY SAP) FOR MASTER DATA ILM OBJECTS (CA_BUPA, FI_ACCRECV, FI_ACCPAYB, FI_ACCKNVK, SCMB_LOC)

- Time reference:
 - `START_RET_DATE` – “Initial Start of Retention”
- Condition fields:
 - `APPL_NAME` – “Application Name”
 - `APPL_RULE_VARIANT` – “Application Rule Variant”
- further cBP or ERP customer/vendor or SCM location specific fields
- Values are determined based on the SoRT information provided by dependent applications during EoP check
- Support policy category `RST` “Residence Rules” used with audit area `BUPA_DP` for the EoP check
- Support policy category `RTP` “Retention Rules” used for destruction within ILM retention management

7.5.2 IRM_CUST FOR ILM OBJECTS PROVIDED BY THE BUSINESS PARTNER DEPENDENT APPLICATION

- use their application specific time references
- use their application condition fields, possible reuse for mapping to application rule variants
- need only to support policy category `RTP` “Retention Rules”

7.5.3 IRMPOL (TO BE MAINTAINED BY SAP CUSTOMERS)

Policies to be maintained (in logical order)

1. Retention periods for all application specific ILM objects relevant for the business of a customer
2. The same retention periods for the master data ILM objects corresponding for all application names and (optionally) application rule variants, which refer to the application specific ILM objects
3. Residence periods for the master data ILM objects and audit area `BUPA_DP` for all application names and (optionally) application rule variants where required.

7.5.4 IRM_CUST_CSS (TO BE MAINTAINED BY SAP CUSTOMERS)

The transaction `IRM_CUST_CSS` provides the maintenance of ILM rule groups. These are to be maintained by SAP customers, if an application decided and implemented this as mapping possibility of their application specific condition fields to application rule variants in their EoP check functionality. So the ILM rule groups provide the following benefits:

- enables in the EoP check of an application to determine relevant application rule variants for the existing data
- provide logical assignment for customers to maintain identical retention periods in master data ILM objects
- original scope: simplify time period maintenance for assigned policies in transaction `IRMPOL`

The explanation how to use ILM rule groups for the EoP check of business partners can be found in chapter "Mapping within EoP Check Implementations using ILM Rule Groups".

8 INTEGRATION SCENARIOS

8.1 INTEGRATION OF THE EOP CHECK FOR CENTRAL BUSINESS PARTNER AND THE ERP CUSTOMER/VENDOR

The following mapping/synchronization possibilities exist:

- CVI ("Customer Vendor Integration")
 - Used to synchronize ERP Customer and Vendor data with business partner data in the local and other systems.
 - The corresponding design can be found here.
- CRM specific mapping
 - The ERP Customer and Vendor functionality can be directly synchronized with the central Business Partner functionality in a CRM system.
 - The corresponding design can be found here.
- SCM specific mapping
 - EoP check for SCM locations which were integrated from ERP is triggered during EoP check of corresponding ERP customer or vendor. It is realized via registration of application `SCM_LOC` and class `CL_SCM_LOC_EOP_CHECK` in ERP Customer and Vendor EoP check customizing.

- The same way EoP check for business partners created during integration of ERP customer of vendor to SCM is triggered.
- ID based mapping in IS-Utilities
 - The synchronization of ERP Customer and Vendor functionality with business partners based on the same ID as in special industry solutions like IS-Utilities.
 - Planned is the implementation via a BAdI provided in the CVI mapping functionality.
- Others (ALE distribution, CIF)
 - The synchronization of ERP Customer and Vendor functionality is possible e.g. via ALE with other ERP systems or non-ERP systems (e.g. using IDoc's of message type DEBMAS to synchronize with BPs in the Auto ID functionality of an SCM system).

8.2 DISTRIBUTION OF THE BLOCKING INFORMATION

Distribution of blocking information to all connected applications and systems:

8.2.1 CENTRAL BUSINESS PARTNER DATA (CBP)

- To block the Business Partner in master system, the function module `BUPA_XPCPT_SET` is used.
- To replicate the same to the dependent systems, the RFC enabled function module `BUPA_PURPOSE_COMPLETE` is used. In parallel a corresponding web service method exists.
- Blocking of existing addresses of a business partner via function module `ADDR_XPCPT_SET`.
- Possibility to set further application specific blocking information via method `BUPA_PURCOMPL_EXPORT` of the BAdI `BUPA_PURPOSE_EXPORT`.

8.2.2 ERP CUSTOMER/VENDOR

- To block data in master system is done via class `CVP_CL_BLOCK_PARTNER`.
- To replicate the same to the dependent systems, the existing ALE message types `CREMAS` (vendor) and `DEBMAS` (customer) are used. In parallel a corresponding enterprise service method exists.
- In addition the BTE events 1321 and 1421 provide also provide the blocking information to registered applications.
- Blocking of existing addresses of ERP customers, vendors or contact persons via function module `ADDR_XPCPT_SET`.
- Possibility to set further application specific blocking information via method `PARTNERS_PURCMPL_EXPORT` of the EoP check interface `CVP_IF_APPL_EOP_CHECK` (part of the registered application specific EoP check class).

8.2.1 SCM LOCATION

- Blocking of SCM location data is done via class `/SCMB/LOC_CL_BLOCK_PARTNER`.

- For SCM location integrated from ERP, blocking in SCM is trigger via RFC function module /SCMB/LOC_BLOCK_HANDLING. This function module also blocks business partners which were created in SCM during integration of ERP customer of vendor
- Possibility to set further application specific blocking information via method PARTNERS_PURCMPL_EXPORT of the EoP check interface /SCMB/LOC_IF_APPL_EOP_CHECK (part of the registered application specific EoP check class).

9 COPYRIGHT AND DISCLAIMER

© Copyright 2011 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials.

The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and

trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

severe malfunctions and is therefore expressly prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP's Support Services and may not be modified or altered in any way.