SAP R/3 Gateway

SAP R/3 Gateway Documentation

Version 2.3.1.18

July 2015

This document applies to SAP R/3 Gateway Version 2.3.1.18.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: SRG-PAGES-231-20150710

Table of Contents

Preface

Software AG's SAP R/3 Gateway is a middleware product that allows you to integrate SAP R/3 with enterprise applications, enabling communication between your application's functions and SAP ABAP functions.

This documentation explains

- how to install Software AG's SAP R/3 Gateway and configure the communication environment
- how to program Software AG's SAP R/3 Gateway components for bidirectional communication
- how to deploy and run the communication components

This documentation addresses IT administrators and application developers who understand the concepts of middleware and are familiar with the webMethods EntireX/Natural environment and/or have some experience in COBOL programming.

The documentation of Software AG's SAP R/3 Gateway is organized under the following topics:

Read Me	Technical product information.
Release Notes	Explains the changes to this version over the previous version.
Introduction	Introduces SAP R/3 Gateway as an integration product in your application environment.
Integration Scenarios	Explains how SAP R/3 Gateway can best be integrated into your application environment
Installation and Configuration	Describes the steps required to install SAP R/3 Gateway and configure your environment.
Programming and Running SAP R/3 Gateway	Illustrates how to generate communication kernels, how to set runtime parameters and how to deploy the kernels.
Using the System Manager	Describes the web application you use to manage SAP R/3 Gateway.
Deploy Kernels	Maintains different environments.
Worker	Background worker tools.
Optimization	Optimization tools.
Configuration	Configuration tools.
Exchange documents asynchronous	The complete IDoc XML Gateway documentation with overview, installation and configuration.

I Readme for Software AG's SAP R/3 Gateway

1 Readme for Software AG's SAP R/3 Gateway

```
Software AG
Uhlandstrasse 12
D-64297 Darmstadt
Germany
Telephone: +49-6151-92-0
Fax: +49-6151-92-1191
E-mail: documentation@softwareag.com
WWW: www.softwareag.com
```


Welcome to Version 2.3.1

Users are encouraged to read the **documentation** before using this product. Documentation is provided in the `doc` subdirectory on the installation CD and online in the web application after **installation**.

Migration

This version contains its own medium to update the previous version 2.2.1 to 2.3.1. The ZIP file medium is on the CD in the sub-directories *windows* and *unix* with the filename *sapr3gatewayUpdate23100.zip*. Please read the section **Migrate to 2.3.1** for information on how to update your installation.

Direct migration from 2.1.1 to 2.3.1 is not possible. To migrate from 2.1.1 to 2.3.1, read the section **Migrate from 2.2.1** and retrieve the 2.2.1 CD-ROM with *sapr3gatewayUpdate221RC7.zip*.

Supported Platforms

This version is released for the following platforms:

- Windows 2003 Standard Edition (32 bit) and Enterprise Edition (32 bit)
- Windows XP Professional
- Windows Server 2008 (64 Bit)
- Sun Solaris 9 and 10 (64 bit)
- SuSE Linux Enterprise Server 10 for x86 (32 bit)
- SuSE Linux Enterprise Server 10 for x86 (64 bit)
- SuSE Linux Enterprise Server 11 for x86 (64 bit)
- SuSE Linux Enterprise Server 10 for IBM zSeries (64 bit)
- Red Hat Enterprise Linux AS 4 and 5 for x86 (32 bit)

- Red Hat Enterprise Linux AS 4, 5 and 6 for x86 (64 bit)
- Red Hat Enterprise Linux 5 for IBM zSeries (64 bit)
- AIX 5.3 (64 bit)

General Dependencies and Software Requirements

To use the SAP R/3 Gateway without IDoc XML Gateway there are no additional software requirements.

For **all platforms**, webMethods EntireX 7.3 or higher webMethods EntireX 8 is supported and can be used.

Update 2.3.1.05

Replaced by **Update 2.3.1.06**

Update 2.3.1.06

Replaced by **Update 2.3.1.07**

Update 2.3.1.07

Replaced by **Update 2.3.1.08**

Update 2.3.1.08

Replaced by **Update 2.3.1.09**

Update 2.3.1.09

Replaced by **Update 2.3.1.10**

Update 2.3.1.10

Replaced by **Update 2.3.1.11**

Update 2.3.1.11

The SAP R/3 Gateway CD labeled 2.3.1.11 contains a new build of *sapr3gateway.war* and *sapr3idocxmlgateway.war*

Update this build by installing **2.3.1.12**.

Update 2.3.1.12

Replaced by **Update 2.3.1.13**

Update 2.3.1.13

Replaced by **Update 2.3.1.14**

Update 2.3.1.14

Replaced by **Update 2.3.1.15**

Update 2.3.1.15

Replaced by **Update 2.3.1.16**

Update 2.3.1.16

Replaced by **Update 2.3.1.17**

Update 2.3.1.17

Replaced by **Update 2.3.1.18**

Update 2.3.1.18

Download this update from **Empower** and extract the compressed files. The changes are described in the **Release Notes** and in detail in section **What's New in 2.3.1.18**. To install this update for existing applications, **upload** the following file:

- *sapr3gatewayUpdate23118.zip* for the manager.
- *sapr3idocxmlgatewayUpdate23118.zip* IDoc XML Gateway.

You can use this update if you are running version 2.3.1.05 or higher.

If you are using webMethods EntireX 8.1, the fix of EntireX RPC Runtime version 8.1.2 patch level 04 build 3 (or higher) is required.

Tip: The webapps installation directory on Software AG's Common Tomcat in conjunction with webMethods EntireX 8.2 can be found at *$SAG_HOME\profiles\CTP\workspace\webapps*. For installation, copy the war files (*sapr3gateway.war*) into this directory.

Tip: To reach Software AG's Common Tomcat from browser clients, port 10010 must be opened in the Windows Server firewall.

Note: SAP has defined an end of maintenance date for the classic RFC SDK and library which are used by SAP R/3 Gateway. Please read SAP note 413708 for more information on **SAP Service Marketplace**. We will introduce our migration strategy to accommodate this SAP notification in due course.

Note: The JAR libraries `activation.jar` and `mail.jar` are no longer part of the shipment. Please download these libraries from the Oracle homepage and place them into the `WEB-INF/lib` directory.

II

2 Release Notes

These Release Notes contain last-minute installation and configuration hints. Please read the **Readme** for more technical information.

Stopping RPC Server

A new function has been implemented for the Rpc2Rfc kernel to deregister this process on webMethods EntireX Broker. Deregistration runs when this process receives a signal `SIGTERM`. To activate the signal handler, you must set the environment variable `EXX_SERVER_ADDRESS`.

The signal handler is implemented in the shared library. The server loads this library at the first RPC call and the signal handler becomes active.

Currently this feature has only been tested on Sun Solaris.

To deactivate the new source code, you can set the define statement `-D EXX_GATEWAY_NO_SHUTDOWN` in makefile.

About Hyperlinks

In some cases, the documentation points to the SAP R/3 Gateway web application in this format: *http://YourGateway:8080/sapr3gateway/manager/index*. Copy this URL, paste it into your browser's address line and change the host name *YourGateway* to yours.

Deinstallation

To remove the installed SAP R/3 Gateway web application, use the Manager of the Application Web Server. With the **Remove** link, the Tomcat Manager (or *http://YourGateway:8080/manager/html/list*) will remove the complete installation from the file system immediately without requiring confirmation. (Please check this if you have a version of Tomcat older than 4.1.29.)

What's new in 2.2.1

- Documentation
- New Configuration
- COBOL Client Generation
- Easy Navigation

- New Upload Feature

Documentation

Several sections have been added to the SAP R/3 Gateway documentation:

- **About Hyperlinks**
- **Deinstallation**
- **Worker**
- **Optimization Tools**
- **Configuration of System Manager**

New Configuration

Some parameters were fixed in makefiles. Now with version 2.2.1, these parameters are defined in the System Manager. This has the following advantages.

- There is one central configuration base.
- There are fewer configuration steps in makefile.
- There is a GUI component.

The following new parameters are transferred by an environment variable from System Manager to the Makefile when the **Compile and Link** job is started.

- The directory of RfcSdk installation
- The directory of webMethods EntireX installation in Windows platform
- The IDL filename
- The library name used in the IDL
- The asynchronous library name used in the asynchronous IDL
- The executable program name of Rfc2Rpc kernel

If you have an older version, please read the migration steps in the section **Migrate to 2.2.1** to use this feature.

COBOL Client Generation

This version contains new features for COBOL client generation. The COBOL client acts as an RPC client for the Rpc2Rfc kernel and with a local sub-program calling convention to call an ABAP business function. When generating the COBOL client stub, it is possible to generate a ready-to-start program. See the section **Generate Stub for COBOL Client** for more information about installation and configuration.

Easy Navigation

The *webapps/sapr3gateway* directory contains an *index.html* file. This file redirects you to the index main menu of the web application. It is now possible to go directly to the web application with URL *http://YourGateway/sapr3gateway* or from the Tomcat Manager web application list.

However, with this feature, the **Browse File System** tool will not work. If you want to continue to use the Browse File System tool, delete or rename the file *index.html*.

New Upload Feature

The new upload and setup feature allows you to transport single files or ZIP files from your desktop to the SAP R/3 Gateway web application. This feature will be used by Software AG's support center to install updates or patches.

For additional information refer to the sections **Migrate to 2.2.1** and **Upload and Setup**.

Migrate to 2.2.1

This section describes how to update from version 2.1.1 to version 2.2.1. If you have a 2.1.1 installation and you have made any changes, it is possible to migrate and to save your configuration.

To migrate from version 2.1.1 to version 2.2.1

1. Back up the current web application (refer to the section **Backup**). The backup tool starts a shell script to collect all the files in the directory and subdirectories *webapps/sapr3gateway*.
2. Copy the update archive 2.2.1 to *webapps/sapr3gateway*.
3. Decide whether you want to update the makefiles (see the section **New Configuration** under **What's New in 2.2.1**). The next 2 steps are optional. In both cases, the update medium does not contain any IDL file.
4. If you wish to update the makefiles, extract the whole update archive with the following command in directory *webapps/sapr3gateway*:

```
jar -xvf sapr3gatewayUpdate221RC7.zip
```

5 If you do not wish to update the makefiles, extract the update archive with multiple commands. The following list includes each file or directory and excludes the makefiles.

```
jar -xvf sapr3gatewayUpdate221RC7.zip ShellScriptSrv.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip AttachManager.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip brokerHWM.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip deployRpc2RfcToDev.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip deployRpc2RfcToInt.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip deployRpc2RfcToProd.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip dev.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip devAdmin.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip devPMQClient.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip docu.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip frameAdmin.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip GASServer
jar -xvf sapr3gatewayUpdate221RC7.zip index.html
jar -xvf sapr3gatewayUpdate221RC7.zip jobs.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip parameter.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip resourceAdmin.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip rpcPing.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip SystemConstancy.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip scheduler.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip setupWizard.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip update.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip version.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip configSetup.xsl
jar -xvf sapr3gatewayUpdate221RC7.zip Rfc2Rpc/dev/revision.xml
jar -xvf sapr3gatewayUpdate221RC7.zip Rfc2Rpc/dev/rfcsrv.tpl
jar -xvf sapr3gatewayUpdate221RC7.zip Rfc2Rpc/dev/rfcstd.c
jar -xvf sapr3gatewayUpdate221RC7.zip Rpc2Rfc/dev/rfctab.c
jar -xvf sapr3gatewayUpdate221RC7.zip Rfc2Rpc/dev/cobol.mak
jar -xvf sapr3gatewayUpdate221RC7.zip Rfc2Rpc/dev/cob_ftp.tpl
jar -xvf sapr3gatewayUpdate221RC7.zip Rpc2Rfc/dev/CobolServer.tpl
jar -xvf sapr3gatewayUpdate221RC7.zip META-INF
jar -xvf sapr3gatewayUpdate221RC7.zip WEB-INF/lib
jar -xvf sapr3gatewayUpdate221RC7.zip Natural
jar -xvf sapr3gatewayUpdate221RC7.zip PMQServer
jar -xvf sapr3gatewayUpdate221RC7.zip PerformanceMeasuring
jar -xvf sapr3gatewayUpdate221RC7.zip Ping
jar -xvf sapr3gatewayUpdate221RC7.zip setup
jar -xvf sapr3gatewayUpdate221RC7.zip docu
```

6 After extraction it is necessary to restart the System Manager. Do this with the Tomcat manager *http://YourGateway:8080/manager/html/list*) **Stop** and **Start** buttons for the web application SAP R/3 Gateway.

7 Some of the parameters in the previous version of the makefiles are placed in the configuration of System Manager. Therefore, the parameter IDL Library must be added to the Development Configuration.

- In the Rpc2Rfc (SAP Server) dialog under **Development** on the **Configuration** menu (or *http://YourGateway:8080/sapr3gateway/manager/devAdmin?cfg=1*)
  - set the value for **IDL Library** to `R3RFC`.
- In the Rfc2Rpc (SAP Calls External Server) dialog under **Development** on the **Configuration** menu (or *http://YourGateway:8080/sapr3gateway/manager/devAdmin?cfg=2*)
  - set **IDL Libarary** to `RFCRPC`
  - set **IDL Asynchronous Library** to `PMQ`
  - set **Make Result** to `rfcrpc.exe` on Windows or `rfcrpc` on UNIX

8 Check the last step and recompile all kernels on the **Development** option.

9 To install the new upload feature, the *sapr3gateway/WEB-INF/web.xml* (*http://YourGateway:8080/sapr3gateway/manager/dir?readfile=$SM_HOME/WEB-INF/web.xml&EditFileDescription=web.xml&edit=*) file must be changed as follows.

1. Add this `servlet` XML node:

```
<servlet>
    <servlet-name>setup</servlet-name>
    <servlet-class>FilterServlet</servlet-class>
    <init-param>
        <param-name>config</param-name>
        <param-value>/configSetup.xml</param-value>
    </init-param>
</servlet>
```

2. Add this `servlet-mapping` XML node:

```
<servlet-mapping>
  <servlet-name>setup</servlet-name>
  <url-pattern>/setup/*</url-pattern>
</servlet-mapping>
```

3. Restart the SAP R/3 Gateway web application.

What's New in 2.2.1.01 - 05

Up to release 2.2.1 some updates are in Software AG's global extranet **Empower**. The following files have been changed and updated.

File	Description
Rpc2Rfc/dev/rfc_c.tpl	The limitation of 32 import, export or table parameters has been removed.
Stylesheet brokerHWM.xsl	Generate error message in system log if SMTP problems occur. The main body contains text about the HWM reached. Set correct log level for system log.
WEB-INF/lib/SystemManagement.jar	Detect and suppress (simultaneously) delete/save operation in configuration file. Suppress ArrayIndexOutOfBoundsException Access Control List maintenance. File (or pipe) handles are closed explicitly after task controller and job controller are ended.
Rpc2Rfc/dev/cob_ftp.tpl	Support of SRVI parameter.
Rpc2Rfc/dev/cleanup.sh and Rfc2Rpc/dev/cleanup.sh	Display a list of deleted files and delete header (.h) files.
Rpc2Rfc/dev/nat_trans.tpl and Rpc2Rfc/dev/nat_prog.tpl	Generate the AD attribute for each parameter in CALLNAT statement. It is now possible to use COMPR=1 in Natural parameter module.
docu/html	The documentation is now available in PDF format. The search engine (SEARCH button) and navigator (CONTENTS button) are also included.
Stylesheet UOWs.xsl	Show only accepted UOWs.
Rfc2Rpc/dev/rfcsrv.tpl	Remove endless loop after sending too much table data.
Stylesheet rpcPing.xsl	Set correct log level for system log and send e-mail notification if RPC communication error occurs.
Rfc2Rpc/dev/abap_clt.tpl	Change the evaluation of IMPORTING, EXPORTING and TABLE parameters on function call.

What's New in 2.3.1

- IDoc XML Gateway Documentation
- Upload with Backup
- Rfc2Rpc Table Support
- Interface Development

- Tomcat 4.1.31

IDoc XML Gateway Documentation

The IDoc XML Gateway documentation contains all configuration, concepts and implementation steps (see the section **IDoc XML Gateway**). In addition, the business concepts of using the SAP R/3 Gateway in various environments are explained in the separate section **Integration Scenarios**.

Upload with Backup

An optional backup of existing files has been added to the upload feature (see the section **Upload and Setup**). Before an existing file is overwritten, it is saved in a ZIP container.

Rfc2Rpc Table Support

The handling of tables has been changed in the Rfc2Rpc kernel if the number of records is greater than defined in the IDL (see the section **Support of Tables**).

Interface Development

Examples have been added to the documentation for IDL and interface development for Rpc2Rfc (in **Calling Scenarios**) and Rfc2Rpc (in **Develop IDL**).

Tomcat 4.1.31

The CD contains the new Tomcat version 4.1.31 to suppress the **Xalan and Ant problem**.

Xalan and Ant

Tomcat 4.1.30 delivers a newer Ant version than Xalan 2.4 requires. If the following symptom appears on your installation, follow the instructions under "resolution".

Symptom:

The menu **Version Info** or the HTTP command *http://YourGateway:8080/sapr3gateway/manager/xalan* Version throws an exception with the message: `java.lang.NoClassDefFoundError: org/apache/tools/ant/launch/AntMain`.

Resolution:

Download Ant 1.5.4 from archive *http://archive.apache.org/dist/ant/binaries* and install *ant.jar* in *$TOMCAT_HOME/common/lib* or delete *ant.jar* in *$TOMCAT_HOME/common/lib*. (This option should only be selected if Ant is not used by any other web applications. Ant is not used by the SAP R/3 Gateway.) The change will take effect after Tomcat has been restarted.

Migrate to 2.3.1

It is possible to migrate from version 2.2.1 of Software AG's SAP R/3 Gateway to version 2.3.1. The CD contains a ZIP file *sapr3gatewayUpdate23100.zip* which has all of the changes between these two versions. The **patches and hotfixes 01-05** are also included. The following files are not on the update medium:

- IDL files
- **System Manager** configuration file *config.xml*
- *WEB-INF/web.xml*

To migrate from version 2.2.1 to version 2.3.1

1. Backup the current installation (with the backup tool, for example; see **Backup**).
2. Upload the ZIP file *sapr3gatewayUpdate23100.zip* with the **Upload and Setup** tool.
3. Restart the web application *sapr3gateway*.
4. Migration to 2.3.1 is now complete. Continue with installation of the optional component IDoc XML Gateway (see the section **Installation**).

Tip: It is possible to install the new SAP R/3 Gateway alongside the previous one in the web application server. To do this, rename the WAR file (e.g. add a version number to *sapr3gateway*) before completing the deployment process.

Ping Wizard supports IDoc XML Gateway Adapter

The **Ping Wizard** has switched to the new IDoc XML Gateway Adapter. Therefore, the old web applicaton *exxr3xmlgateway.war* is no longer delivered. The stylesheet *ping* supports the old adapter. You do not need to make any changes to the timer task command (see the section **Scheduler**). Call the **Ping Wizard** stylesheet to make any changes to the new adapter, after which you can remove the old one.

Migrate EntireX Communicator

If you upgrade your EntireX Communicator installation on UNIX from v6 to v7 or v71 to v72, the UNIX setup will place the software in its own version directory and create a new `sagenv` script.

To change only the runtime for the running kernel tasks (**Rpc2Rfc** and **Rfc2Rpc**), you can set the version directory of EntireX (**EXX_VERS**) on the **System Constants page**. After this has been set, you must restart the kernels.

Tip: After restarting the kernels, this Rpc2Rfc kernel version writes to the **System Log**. Alternatively, set the environment variable `ETB_STUBLOG=1` (**Rpc2Rpc** and **Rfc2Rpc**) and check the created file for version information. This file is created in the same directory as the running task with the process ID and extension `etb`. This step makes sure that the new EntireX runtime will be used. Do not forget to undo this trace setting afterwards.

To recompile the kernels

1. Change the EntireX version directory **EXX_VERS** on the **System Constants page**.
2. Set the new `sagenv` script **SAG_ENV_SCRIPT** on the **System Constants page**.
3. Delete all old generated source code on the **Jobs page** by executing **Rpc2Rfc clean up Dev files** and **Rfc2Rpc clean up Dev files**.
4. Go to the **Developer page** and execute **Compile and Link** for Rpc2Rfc and Rfc2Rpc.
5. Go to the **Running Task page** and stop all tasks in the development environment.
6. Go to the **Deployment page** and copy all new generated kernels into the run directory.
7. Start the kernels.

Datatype I4

On 64 bit platforms, a problem can occur with data type `I4` in the **Rfc2Rpc kernel**.

Symptom:

The ABAP clients receive the exception EXX00010083. (The **System Log** also displays the message `ERX Call stub: Parameter out of value space. Location of wrong Parameter...`).

Resolution:

We recommend replacing the data type `I4` by data type `P`. Please contact Software AG support if you cannot use this solution.

Platform Encoding

The XML-RPC servlet, a component of IDoc XML Gateway (**IDoc type generation**), needs the platform character encoding (default character encoding) setting of JVM.

Symptom:

During **IDoc type generation** in IDoc XML Gateway, the System Log displays the error message `Broker Error 0022 0437: Some chars unconvertible to target CP`.

Analyze:

To see and trace the character encoding used, set and activate the following **JVM properties**.

```
entirex.sdk.default.trace.level=STANDARD
entirex.sdk.default.trace.filename=STDOUT
```

1. After activating the JVM properties, restart the sapr3idocxmlgateway web application
2. Generate the IDoc type.
3. In Tomcat, the trace is written to *$TOMCAT_HOME/logs/catalina.out*.

4. Analyze the log for `encoding` print outs.

Resolution:

Set the **JVM property**:

```
file.encoding=To_Your_Need__Platform_Encoding
```

After the property has been activated, restart the sapr3idocxmlgateway web application.

Microsoft C Compiler

Symptom:

When compiling the Rpc2Rfc server or the RfcIdl tool, the Microsoft Windows C-Compiler aborts with error message `fatal error C1902: Program database manager mismatch; please check your installation`.

There is a compiler option `/FD` to generate `.pdb` files for debugging information.

Resolution:

Delete the `/FD` compiler option in the makefiles *Rpc2Rfc/dev/MakefileNT* and *RfcIdl/MakefileNT*.

What's New in 2.3.1.02 - 03

This documentation has a new **pipeline view** section. In detail, the following files have been updated.

File	Description
setupWizardIDocXMLGW.xsl	Misspelled words have been corrected.
brokerHWM.tpl	Request response time is now displayed.
Rfc2Rpc/dev/abap.tpl	IN and OUT groups are now supported. This is mapped to one IMPORTING and one EXPORTING parameter. The results and contents of a table after CALL FUNCTION are displayed.

File	Description
Rfc2Rpc/dev/rfcstd.c	The endless loop that occurred if wrong parameter was passed has been removed.
Rfc2Rpc/dev/rfcsrv.tpl	I4 parameter data type on 64-bit systems is now supported. To activate the last 3 changes, compile and link the Rfc2Rpc kernel.
pipeline.xsl	The wrong index to edit the Attach Manager and indent child pipeline has been corrected.
scheduler.xsl	Text area for commands. The sequence of all timer tasks is fixed.
schedulerResponseEx.xsl	Response of scheduler exception is now displayed.
Rpc2Rfc/dev/nat_trans.tpl	The suffix 'P' for test programs has been corrected. The programs RFCINTFB, RFCINTPA, IDOCINBD and IDOCTYPE are filtered and not passed to the SYSTRANS file. Additional **parameters** are supported to suppress the subprograms.
RfcIdl/rfcidl.c	ABAP 'Table Types' are now supported. The ABAP programmer can define a table parameter as table type. The RfcIdl tool now retrieves the structure definition from the SAP dictionary. To activate the new feature, compile the RfcIdl tool on **Setup Wizard**.
deployRunningTask.xsl	This is a new tool to perform an easy and smart deployment. The feature is described on the **deployment** page. The first call of this page with *http://YourGateway:8080/sapr3gateway/manager/deployRunningTask* creates its own menu item labelled **Smart Deployment** under **Tools**.
IDoc Gateway/IDocType_ToFile.xsl	Misspelled words have been corrected.
Natural/PMQClient/NatPMQClient.tpl	Natural objects are not able to compile if FTP is used for upload. The tabulator characters are deleted. Incorrect field names generated, if level greater than 2.
IDoc Gateway/IDocStd.xsl	'rpc-broker' and 'rpc-service' are passed to the stylesheet. This allows you to select an RPC server on a second broker.
IDoc Gateway/IDocType_ToIDL.xsl	All fields are now generated for data type 'A'. Data type 'N' is no longer supported because some 'N' fields must be transported with spaces; this is not possible because Natural initializes an 'N' field with '0'.
acl.xsl	Misspelled words have been corrected.
setupWizardIDocXMLGWpipeEnv.xsl	Environment parameter for Attach Manager creation has been added.
setupWizardIDocXMLGW.xsl	An additional environment parameter for Attach Manager creation has been added. Depending on the selected environment, the **RPC service** is set. There is a new development environment to develop **IDoc Outbound XML RPC** services. The new wizard step can create this environment.
docu.xsl	Support of more pages for context-sensitive help.

File	Description
SystemConstancy.xsl	This page contains a lot of parameters for configuration. The name is changed to **System Constants** on the user interface and documentation.

What's New in 2.3.1.04

This update also contains the files of the **previous fix**. In detail, the following files have been updated.

File	Description
WEB-INF/SystemManagement.jar	■ Support of **Xalan** 2.5.4 or 2.6. ■ The timer task events are evaluated correctly now in all time zones. ■ The **attach manager** supports a new synchronization model. ■ New **Queue Service Info** **pipeline** step is supported for quicker response. ■ Solves a problem starting Rpc2Rfc or Rfc2Rpc kernel in a WebSphere environment. **Note:** Depending on the web application server, you must restart the SAP R/3 Gateway and IDoc XML Gateway immediately after **upload** or the restart is automatically done.
acl.xsl, brokerHWM.xsl, devAdmin, files.xsl, frameAdmin, jobs.xsl, links.xsl menuAdmin, rpcPing.xsl	Support of WebSphere environment.
scheduler.xsl	Support of time zones.
pipeline.xsl and pipelines.xsl	Support of new pipeline step.
parameter.xsl	Support of the functions Copy a Deployment Environment and Create a New Deployment Environment.
Rpc2Rfc/rfctab.c, rfctab.h, rfc_c.tpl	The logon strategy **Function call overwrites Running Task parameter** has been improved.
Natural/PMQClient/NatPMQClient.tpl	Support of long IDoc segment names for a **Natural client**.
devAdmin.xsl	Support of Copy a Development Environment.
docu/*	The changes are described in this new documentation.
GASServer/main/GASDISP.cob	Support of RPC ping and the server stub is called with leading 'S'.
setupWizard.xsl	Misspelled words have been corrected.
PageHead.xsl	The **JAVA_HOME parameter of System Constants** is passed as second parameter to the backup script.

File	Description
IDoc Gateway/PMQdoc_toRPC.xsl, IDoc Gateway/XMLIDoc_ToCustom.xsl, setupWizardIDocXMLGW.xsl	Support of **Outbound XML-RPC Development**.

What's New in 2.3.1.05

This documentation has a new **Clone Environment Wizard** section. The Clone Environment Wizard makes it possible to create an environment in which multiple IDLs of Rpc2Rfc or Rfc2Rpc kernels can be run in parallel. This update also contains the files of the **previous fix**. In detail, the following files have been updated.

File	Description
WEB-INF/SystemManagement.jar, cloneEnvironment.xsl	The new **Clone Environment Wizard** provides support for more than one project in one SAP R/3 Gateway instance.
packageBuilder.xsl	The **Package Builder** increases deployment. Deployment runs with the HTTP protocol to copy the executable files from multiple application servers.
deployRunningTask.xsl	**Smart Deployment** is described and now supports the **Package Builder**.
Rpc2Rfc/dev/rfctab.c, rfc_c.tpl	Supports data type `F8`. Suppresses compilation errors.
RfcIdl/rfcidl.c	Supports binary data type `B`.
docu.xsl, version.xsl	Supports new stylesheets.
docu/*	The changes are described in this new documentation.
brokerHWM.xsl	The correct request response time is now displayed in the **Broker High Water Marks**.

What's New in 2.3.1.06

The **software requirements** are changed for EntireX 7.2.1. Patch 68 is required.

This update also contains the files of the **previous fix**. In detail, the following files have been updated.

File	Description
AttachManager.xsl, WEB-INF/SystemManagement.jar	Supports new synchronisation type to handle parallel requests in **Attach Manager**.
deployRunningTask.xsl	Supports the shutdown process of Running Tasks if **Shutdown Command URL** parameter is set.
run.xsl	Displays user IDs of server if **Shutdown Command URL** is executed.
dir.xsl	The **Execute** button is generated for `.bat` and `.sh` shell sripts to execute immediately.
SystemConstrancy.xsl, MakefileAIX64GNU	Supports AIX 64 bit platform with GNU C compiler.
rpcPing.xsl	Improved exception handling.
Natural/PMQClient/NatPMQClient.tpl	The generated source has been changed. The connection parameters are settings in *PMQINIT* module. The new interface is described in the **documentation**.
setupWizardIDocXMLGW.tpl, CobolPMQClient/CobolPMQClient.tpl, CobolPMQClient/CobolPMQClientPgm.tpl, CobolPMQClient/make.sh, CobolPMQClient/make.bat and CobolPMQClient/PMQ.idl	Supports the development of a **COBOL client**, which can generate IDocs into the Persistent Message Queue (PMQ).
docu/*	The changes are described in this new documentation. The **Gateway Logon Strategy** has been documented in its own section. Additional information for **Using Tomcat 5**. Externel Links to the documentation of EntireX are changed to version 7.2.1.50
UOWcontroller.xsl	The new version has lower memory resource requirements and can handle many active UOWs.
eventDispatcherAdmin.xsl, mailEventDispatcher.xsl, WEB-INF/SystemManagement.jar	An **Event Dispatcher** has been implemented for the **System Log**. The administrator can define events for notification (e.g. e-mails).
IDoc Gateway/XMLToIDocStd.xsl, IDoc Gateway/XMLToIDoc_FromHTTP.xsl, IDoc Gateway/XMLToIDoc_FromPMQ.xsl	The delivery process through the **inbound pipeline** to SAP has been improved. Performance is better with big IDoc documents.
AttachManager.xsl, **pipelines.xsl**	Supports **Copy to** button.
packageBuilder.xsl	Set the currect user ID and password to login into the target system.
GASServer/dev/CobolServer.tpl	Suppresses compilation error, if no OUT (send) field exists.
Rfc2Rpc.xsl	The dialog supports the settings of Natural RPC Security parameter.
IDoc Gateway/config/AdapterConfig.xml, IDoc Gateway/RFCSTD.xmm, rfcMeasure.xsl	The new `rfcMeasure` page prints out measure points of the Rpc2Rfc kernel. The measure point are collected during the RFC session.

File	Description
deployRunningTask.xsl, cloneEnvironment.xsl	Suppresses the error message `Can not resolve namespace prefix: java` on calling this page.
Rpc2Rfc/dev/xml_rpc.tpl, Rpc2Rfc/dev/xml_rpc.win.mak, Rpc2Rfc/dev/xml_rpc_unx.mak, devAdmin.xsl	The devlopment process supports a **XML RPC client**.
WEB-INF/entxrt.jar	Update to EntireX 7.2.1.50.
Tools/exxshutdown.c	Supports EntireX 7.2. The older compiled version works only with EntireX 7.1. To (re-) compile and link the executable on UNIX, use **Help**, **Setup Wizard** step #9.
PerformanceMeasuring/EXXGateway.xls, PerformanceMeasuring/RFCSTD.dll	*RFCSTD.dll* supports EntireX 7. In *EXXGateway.xls*, parameters are changed.
Rpc2Rfc/dev/rfctab.c, Rpc2Rfc/dev/revision.xml	Correct an error in `GetTickCount()` function for UNIX and suppress a RPC error message on getting measure points.

What's New in 2.3.1.07

This update also contains the files of the **previous fix**. In detail, the following files have been updated.

File	Description
acl.xsl, WEB-INF/SystemManagement.jar	Supports exclude list of resources in the **Access Control List**. The administrator can explicitly disallow the access to special resources.
AttachManager.xsl, WEB-INF/SystemManagement.jar	Displays start and stop time. Restart command is available.
IDoc Gateway/PMQdoc_toXBD.xsl	This stylesheet delivers **Outbound** messages to Service Orchestrator.
encodePW.xsl	Utility to **encode** the saved passwords.
RfcIdl/rfcidl.c	Supports complex export parameters, SAP unicode repository and cuts the long table field names. Supports generation of unique field names for `Offset`, `Fill` and `Count` parameters. To (re-)compile and link the executable, use Help, Setup Wizard.
Rfc2Rpc/ClientThreadedPMQv611.tpl	Supports data type `I4` to generate messages for Persistent Message Queue (PMQ).
index.xsl, searchWorker.xsl	**Searches for worker items** in configuration.
Rpc2Rfc/rfc_c.tpl	Supports data type `I4` for 64-bit systems.
Rpc2Rfc.xsl, **Rfc2Rpc.xsl**	New field for adding parameter `SAP_CODEPAGE`
Rfc2Rpc/dev/aba_clt.tpl	Generates ABAP field names with _ instead of `-` character.

File	Description
Rfc2Rpc/dev/sag-idl.tpl, Rfc2Rpc/dev/aba_clt.tpl, Rfc2Rpc/dev/rfcsrv.tpl	Supports **fixed-size arrays on level 2**.
IDocGW/*.xls, packageBuilder.xsl	Supports new namespace of System Manager. To receive parameters for a stylesheet, the namespace is changed to `com.softwareag.sm.resource.TransformerRequestProperties`.
ping.xsl	Supports the **Ping Wizard** without setting the mail address of Gateway administrator (`MAIL_TO_GW_ADMIN`). The Ping Wizard can be used to cache the RFC handle of **Rpc2Rfc kernel** (parameter `EXX_RFC_TIMELIMIT`) the whole time.

What's New in 2.3.1.08

This update also contains the files of the **previous fix**. In detail, the following files have been updated. The main changes of this update are to support EntireX 7.3 Developer's Kit. The **Setup Wizard** contains addional steps for configuration.

File	Description
Rfc2Rpc/dev/MakefileLINUX	The build process is now starting.
Rfc2Rpc/dev/MakefileSUSEZLINUX64	Compiler `PIC` option is added.
*.tpl, setupWizard.xsl	Support EntireX 7.3 IDL compiler and suppress warning messages.
RfcIdl/rfcidl.c	**RfcIdl** supports new parameter to set number of default table items.
devIDL.xsl	**New IDL editor** for adding and deleting functions.
pipeline.xsl, pipelines.xsl	Supports logon with EntireX security. UOWs can purge for a queue/service.
IDocGW/IDocStd.xsl	Supports logon with EntireX security.
cloneEnvironment.xsl	Suppresses duplicate strings (of new environment) in new created Running Task names.
Rfc2Rpc/ClientThreadedPMQv611.tpl, Rfc2Rpc/ClientThreadedPMQv731.tpl	Supports new parameter for conversation ID handling.
Rpc2Rfc/dev/cob_prog.tpl	Supports trancated field names.
Rpc2Rfc/dev/rfctab.c	Supports addional connection parameter `EXX_RFC_GWHOST`.
docu/*	Switch to Internet documentation of Crossvision EntireX Communicator version 7.3.
upload.xsl	Supports the upload and update of created project **environments**.
build/CSOAdapter4SAPR3.zip	Crossvision Service Orchestrator custom component.

File	Description
build/Rpc2RfcUpdate23108.zip	Contains all changes for the Rpc2Rfc kernel development. **Upload** this file for created project **environments**.
build/Rfc2RpcUpdate23108.zip	Contains all changes for the Rfc2Rpc kernel development. **Upload** this file for created project **environments**.

EntireX Communicator 7.3 on AIX 5.3

Symptom:

When linking the Rpc2Rfc, Rfc2Rpc server or the RfcIdl tool, the linker aborts with the following error message:

```
ld: 0711-317 ERROR: Undefined symbol: .iconv_open
ld: 0711-317 ERROR: Undefined symbol: .iconv_close
ld: 0711-317 ERROR: Undefined symbol: .iconv
```

The character encoding conversion library is missing.

Resolution:

Add the parameter `-liconv` to the linker option in the makefiles. To change the makefiles, go to **guimenu** and **Development**, and select **Makefile** to start the editor.

What's New in 2.3.1.09

This update also contains the files of the **previous fix**. In detail, the following files have been updated.

File	Description
build/CSOAdapter4SAPR3.zip	Avoids problem with field names which cannot use as XML element tag.
Rpc2Rfc.xsl	Optimizes the maintenance of configuration. The external RPC server configuration file can import into the **System Manager**. The Gateway trace parameter is now restricted and selectable in a combo box.

File	Description
run.xsl	Has been optimized for better performance. In addition to the `Log` output command, the UNIX `tail` command is available and shows the last output lines.
Rpc2Rfc/dev/*.c, Rpc2Rfc/dev/*.h, Rpc2Rfc/dev/*.tpl, Rpc2Rfc/dev/revision.xml	Revision 15 of **Rpc2Rfc kernel** no longer prints the RFC password into the system log.
build/PMQConnector.rar	The JCA connector receives EntireX persistent messages and dispatches these to MDB (message driven bean). The API documentation defines the dispatcher and listener interface.
WEB-INF/SystemManagement.jar	Avoids `StackOverflow` in **Attach Manager**.
CobolPMQClient/CobolPMQClient*.tpl	Avoids compiler errors on IBM mainframes.
pipeline.xsl	The **System Log** can be displayed locally and in context with the **Pipeline Step**.
Rpc2Rfc/dev/nat_prog.tpl; Rpc2Rfc/dev/nat_trans.tpl	Generates correct code for array parameters.
WEB-INF/PMQServer.jar	The Java PMQ Stream and Server API implements `prepareCommit()` method to support transactions in distributed environment.
IDocGW/IDocToXMLStd.xls	Suppresses recursive loop on large IDoc types.
IDocGW/IDocToXML_ToPMQ.xsl	Logging information has been added for the transformation process.
devIDL.xsl, WEB-INF/SystemManagement.jar	Supports new `Compare` and `Replace` functions in the **IDL Function Editor**.
RfcIdl/rfcidl.c	Supports field names starting with a numeric character.
cloneEnvironment.xsl	Generates correct path for the IDL file in a new development environment if there are two or more existing Rpc2Rfc environments.

What's New in 2.3.1.10

This update also contains the files of the **previous fix**. In detail, the following files have been updated:

File	Description
build/PMQConnector.rar and build/RPCConnector.rar	Supports message driven beans with these connectors in **pipelines.**
build/RPCConnector.rar	The JCA connector receives EntireX RPC requests and dispatches them to EJB. The API documentation defines the dispatcher and listener interface.

File	Description
Rfc2Rpc/dev/rfcsrv.tpl, Rfc2Rpc/dev/abap_clt.tpl	Supports logical data type `L`, suppresses Compiler warnings and corrects evaluation number of `INOUT` parameter. APAB report generation supports new **data types** and fixed arrays on level 2. **Additional parameter** changes the memory allocation to support big data size.
upload.xsl, WEB-INF/SystemManagement.jar	Supports logon into System Manager if web application security is enabled.
pipeline.xsl, pipelines.xsl, queueReadNextTXT.xsl, queueReadNextXML.xsl, UOWs.xsl	Displays the delivery count of UOWs. Existing pipelines can be reached directly with HTTP Get parameter `?name=<pipeline name>` to the URL. New **pipeline step** is supported for reading the UOW message contents.
WEB-INF/entxrt.jar	Updates EntireX Java runtime.
cloneEnvironment.xsl	Sets library path correctly after cloning a Running Task environment.
PageHead.xsl, menuAdmin.xsl	Supports URL in top menu and in menu item list.
dev.xsl, devAdmin.xsl, devIDL.xsl, devPMQClient.xsl	The development environment can be reached with HTTP parameter `?show=<index>`. New command on **Development administration** allows generation of quick access menu items.
Rpc2Rfc/dev/rfcstd.c, Rpc2Rfc/dev/sag-idl.tpl	The sag-idl template passes the comments of fields from original (generated from SAP) IDL to the EntireX IDL.
Rfc2Rpc/dev/ClientThreadedv80.tpl, Rfc2Rpc/dev/ClientThreadedPMQv80.tpl	Supports webMethods EntireX 8.0.
CobolPMQClient/*.tpl	Supports more data types (`I`, `N`, `NU` and `P`), publish and subscribe protocol and B2000 cobol compiler.
docu/*	Supports links to webMethods EntireX.

What's New in 2.3.1.11

This update also contains the files of the **previous fix**. In detail, the following files have been updated:

File	Description
WEB-INF/lib/mail.jar, WEB-INF/lib/activation.jar	Supports **Tomcat 5**
Rpc2Rfc/dev/MakefileLINUX64, Rfc2Rpc/dev/MakefileLINUX64, RfcIdl/MakefileLINUX64, SystemConstancy.xsl	Supports all **Linux 64 bit platforms**.

File	Description
Rpc2Rfc.xsl	Some parameters can be set directly into the running Rpc2Rfc server by using the **Activate** button. Restarting the server or saving the parameters is not necessary.
pingText.xsl	This stylesheet returns the result of **ping** as text. The output can be used in your monitoring software.

What's New in 2.3.1.12

This update also contains the files of the **previous fix** and supports new **SuSE Linux Enterprise Server 11 platform**. In detail, the following files have been updated:

File	Description
WEB-INF/web.xml	Supports file system browse feature in WebDAV servlet under Tomcat 5. The character * is added in URL pattern. **Note:** After updating and changing this file the Web Application server is restarting SAP R/3 Gateway. To avoid the restart or if you have customized *web.xml*, remove this file from the update ZIP.
DeployRunningTask.xsl	Corrects error in selecting the package if local environment does not exist.
parameter.xsl	Drops links if destination file does not exist.
OutboundXML/make.bat	Supports webMethods EntireX Workbench 8 batch mode.
docu/*	Supports webMethods EntireX 8.1 documentation.
setupWizard.xsl, Rpc2Rfc/dev/unix.mak	Changes are made for webMethods EntireX 8.1.
Rfc2Rpc/dev/ClientThreaded.tpl, Rfc2Rpc/dev/ClientThreadedPMQ.tpl, Rfc2Rpc/dev/MAKEFILE*	Remove webMethods EntireX version information in filename.
*/make.sh, */MAKEFILE*	Change of calling webMethods EntireX IDL Compiler (erxidl).
Rpc2Rfc/dev/rfc_c.tpl, Rpc2Rfc/dev/revision.xml, Rpc2Rfc/dev/rfctab.c	Avoids `type cast pointer` compiler warnings.
Rfc2Rpc/dev/rfcsrv.tpl, Rfc2Rpc/dev/revision.xml	The **EXX_RPC_RETRY_COUNT** parameter is added to control the retries on failed RPC requests.
cloneEnvironmentWizard.xsl, deployRunningTask.xsl, dir.xsl, files.xsl, JavaSrc.xsl, NaturalSrv.xsl, Rfc2Rpc.xsl, Rpc2Rfc.xsl, run.xsl, setupWizard.xsl, ShellScriptSrv.xsl, SystemConstancy.xsl	Internal changes and optimization.

File	Description
build/Rfc2RpcUpdate23112.zip, build/Rpc2RfcUpdate23112.zip	Contains only changes for specific kernel environment. Use the ZIP files to update the kernel environment with **upload** tool.

What's New in 2.3.1.13

This update also contains the files of the **previous fix** and supports the new **Windows Server X 64 Bit** platform in conjunction with webMethods EntireX 8.2. In detail, the following files have been updated:

File	Description
Rpc2Rfc/*.c, Rpc2Rfc/*.h, Rpc2Rfc/*.tpl, Rpc2Rfc/revision.xml, Rpc2Rfc/MakefileWinX64	Rpc2Rfc server supports new Windows Server X 64 Bit platform.
Rfc2Rpc/*.c, Rfc2Rpc/*.h, Rfc2Rpc/*.tpl, Rfc2Rpc/revision.xml, Rfc2Rpc/MakefileWinX64	Rfc2Rpc Server supports new Windows Server X 64 Bit platform.
RfcIdl/rfcidl.c, RfcIdl/MakefileWinX64, RfcIdl/MakefileNT	IDL generation tool supports new Windows Server X 64 Bit platform.
Rpc2Rfc/autoconv.c, Rfc2rpc/autoconv.c, RfcIdl/autoconv.c, Rpc2Rfc/Makefile*, Rfc2rpc/Makefile*, RfcIdl/Makefile*	Library *autoconv.lib* is obsolete and was removed from makefiles because source *autoconv.c* was moved from webMethods EntireX to these SAP R/3 Gateway projects.
Rpc2Rfc/make.bat, Rfc2rpc/make.bat, RfcIdl/make.bat	Batch scripts have been extended to call 64 Bit Visual C-Compiler environment. Check the system path to Visual C-Compiler in **Setup Wizard**.
brokerHWM.xsl	Suppresses graphic visualization when data are not available.
dev.xsl, devIDL.xsl	Supports new Windows Server platform.
JavaSrv.xsl	Supports import of configuration file and editing in own editor.
PageHead.xsl	Performance optimized.
run.xsl	Additional link to working directory is added.
setupWizard.xsl, SystemConstancy.xsl	Supports new Windows Server platform.
upload.xsl	After upload with saving existing files, a download link is available for the compressed old files.
Rpc2Rfc/unix.mak, Rpc2Rfc/MakefileNT	Removes duplicate compiler calls.
build/ManagerUpdate23113.zip	Collects all changed stylesheets (*.xsl) for the user interface.

Upload the full content of *sapr3gatewayUpdate23113.zip* only on new SAP R/3 Gateway. On existing and running nodes, upload the partial ZIP files (e.g. *build/ManagerUpdate23113.zip*) to save your local changes in configuration files (e.g. *Makefiles*).

What's New in 2.3.1.14

This update is only a new build of **13** and contains the changes since **05**. Please read **What's New in 2.3.1.13** for relevant changes.

What's New in 2.3.1.15

This update contains all changes from version **2.3.1.05** to version **2.3.1.13** and fixes the following files.

To support EntireX 8.2.2, this update contains new Unix Makefiles. When the new EntireX runtime is used, the linking step in all Makefiles must be executed because library `libss.so` has been removed.

File	Description
WEB-INF/lib/activation.jar, WEB-INF/lib/mail.jar	The JAR libraries are no longer part of the shipment. Please download these libraries from the Oracle homepage and place into `WEB-INF/lib` directory.
Rfc2Rpc/rfcsrv.tpl, Rfc2Rpc/revision.xml	Data types I1 and I2 are moved to I4 internal to get support from RFC SDK on registering external server.
JavaSrv.xsl	Performance optimized.
WEB-INF/SystemManager.jar	IDL reader supports simple array definition, e.g. (/10).
devIDL.xsl	Corrects exception handling.
Rpc2Rfc/unix.mak	Adds dependencies to compile required objects.
Rpc2Rfc/rfctab.c, Rpc2Rfc/revision.xml	Removes MSHOST length limitation of RFC connection parameter.
Rpc2Rfc/Makefile*, Rfc2Rpc/Makefile*, RfcIdl/Makefile*	Removes linking library references `libss.so` in all Unix Makefiles.
build/Rpc2RfcUpdate23115.zip, build/Rfc2RpcUpdate23115.zip, build/RfcIdlUpdate23115.zip	Contains only changes for specific kernel environment (Rpc2Rfc and Rfc2Rpc) and development tool (RfcIdl).

What's New in 2.3.1.16

This update contains all changes from version **2.3.1.05** to version **2.3.1.15** and fixes the files in the following table.

This update version has been tested with EntireX 9.5 SP1.

File	Description
Rpc2Rfc/rfctab.c	Supports passwords up to 40 characters.
build/WxSRG.zip	This package is required if you plan to migrate from the existing SRG implementation to a webMethods Integration Server environment; for details see **SAP R/3 Gateway Migration**.
devIDL.xsl, build/RfcIdlUpdate23116.zip	Supports new option `webMethods SAP Adapter Calling Convention`.

What's New in 2.3.1.17

This update contains all changes from version **2.3.1.05** to version **2.3.1.16** and fixes the files in the following table.

File	Description
Rpc2Rfc/MakefileWinX64	Missing dependencies are added to generate all of the required source code.
build/WxSRG.zip	Dependency to WxDevTools package is removed. Generation of Rpc2Rfc mapping flow-services is increased; for details see **SAP R/3 Gateway Migration**.

What's New in 2.3.1.18

This update contains all changes from version **2.3.1.05** to version **2.3.1.17** and fixes the files in the following table.

File	Description
build/WxSRG.zip	Generation of Rpc2Rfc mapping flow-services is increased with regard to RFC default tables in the Adapter function.
devIDL.xsl, build/RfcIdlUpdate23118.zip	Supports new option for reporting IDL date, time and documents fields used in Rpc2Rfc mapping flow-services.

3 Introducing Software AG's SAP R/3 Gateway

This chapter introduces you to the underlying concept of Software AG's SAP R/3 Gateway and illustrates its architecture in juxtaposition with SAP R/3 and enterprise application functions.

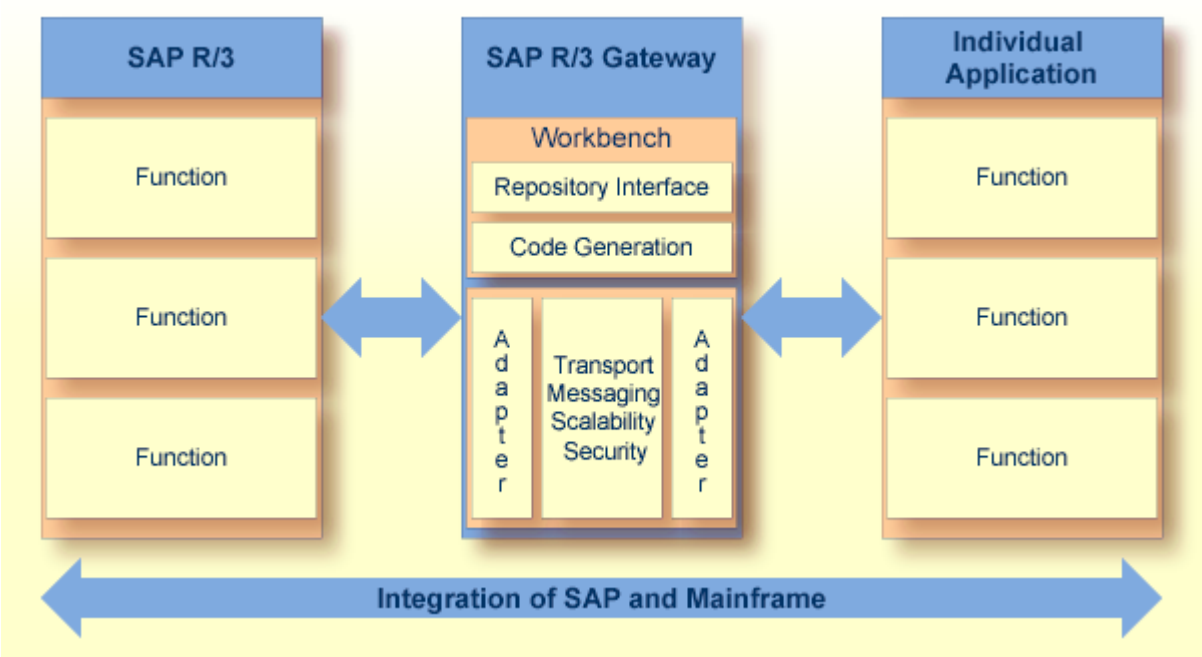
Integration with Software AG's SAP R/3 Gateway

The increasing practice of connecting computers in enterprise networks has given rise to business models that require the integration of applications. The SAP R/3 Gateway enables you to design a service-oriented architecture for your computing environment which supports all technical communication for your business applications.

Using SAP R/3 Gateway, it is easy to program and execute a function call over distributed applications. Once you have understood the principle, designing the corresponding Business Process is a formality. You will see that

- it is easy to call an ABAP function when communication comes from outside SAP R/3.
- it is easy to call your mission-critical application when communication is generated within SAP R/3.

The following figure illustrates how SAP R/3 Gateway links SAP R/3 functions with enterprise application functions.



- SAP R/3 Gateway provides a developer's Workbench in the form of a web application that shows you how to provide all of the parameters required to generate communication code.

- The communication programs (Adapters) translate Remote Procedure Calls (RPCs) to Remote Function Calls (RFCs) for communication from enterprise applications to SAP, and they translate RFC to RPC for communication from SAP to enterprise applications. These communication components are called the Rpc2Rfc and Rfc2Rpc kernels, respectively.
- The communication infrastructure is provided by Software AG's webMethods EntireX technology.

The SAP R/3 Gateway supports both synchronous and asynchronous integration of your business application. The section **Integration Scenarios** explains which is the best one and how each can used.

Why Use Software AG's SAP R/3 Gateway

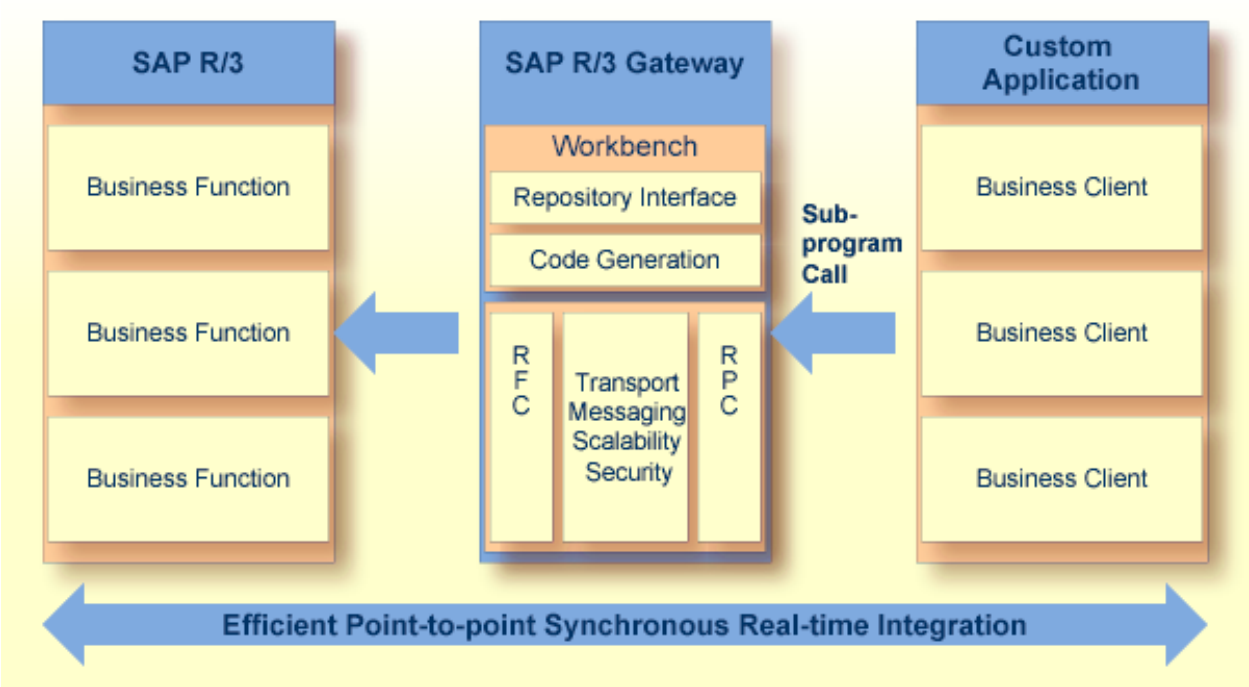
The benefits of using Software AG's SAP R/3 Gateway can be summarized as follows:

- SAP R/3 Gateway is platform independent in both the development and production phases.
- When working with SAP R/3 Gateway, every step - both in the development environment and in the production environment - is defined within the browser application. This means a portal is provided to manage the entire integration project.
- SAP R/3 Gateway can be scaled to meet your IT requirements.
- As administrator or application developer, you do not need any specialist knowledge about the operating environment for communication.
- The deployment process of communication components supports version control of the development, integration (quality) and production environments.

4 Integration Scenarios

Custom Application calls SAP Function

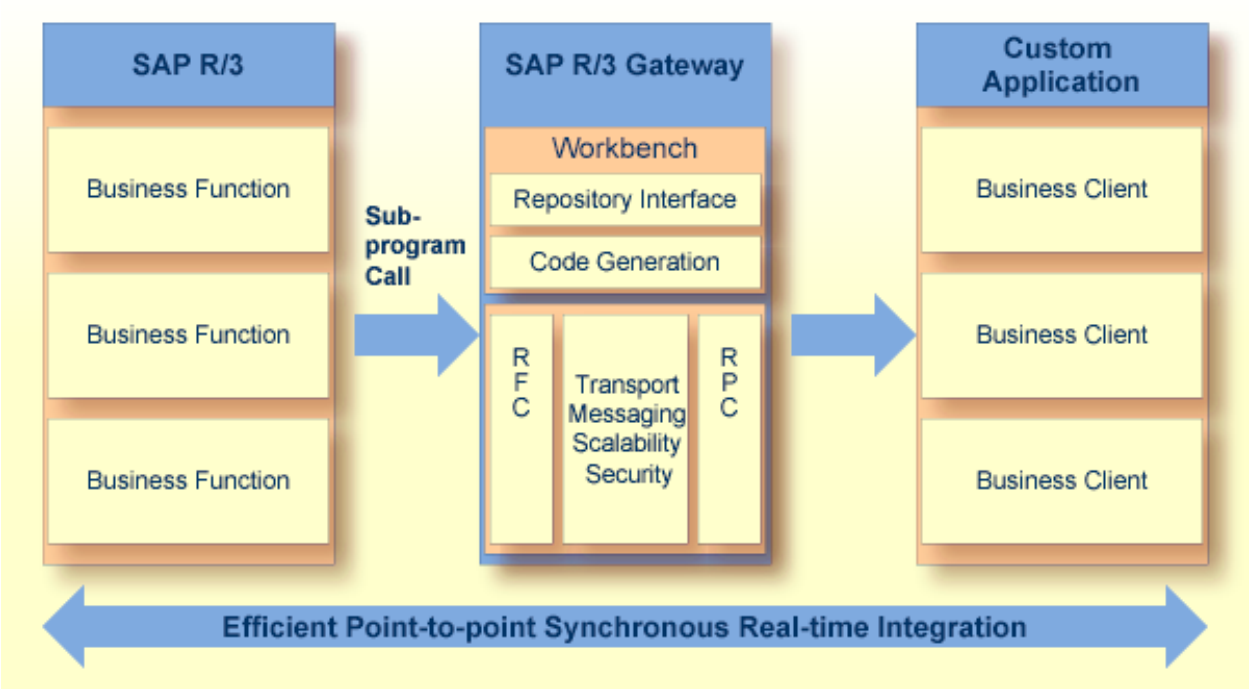
This synchronous integration scenario makes it possible to call an SAP R/3 business function from your mission-critical application.



The SAP R/3 business function is implemented as a remotely enabled ABAP function. You can call a standard BAPI or your own implemented RFC function. In either case, the development kit in this scenario supports the generation process to retrieve the business interface for your mission-critical application. See the section **Develop a Client Call to SAP/R3 (Rpc2Rfc Kernel)**.

SAP calls Custom Application Server

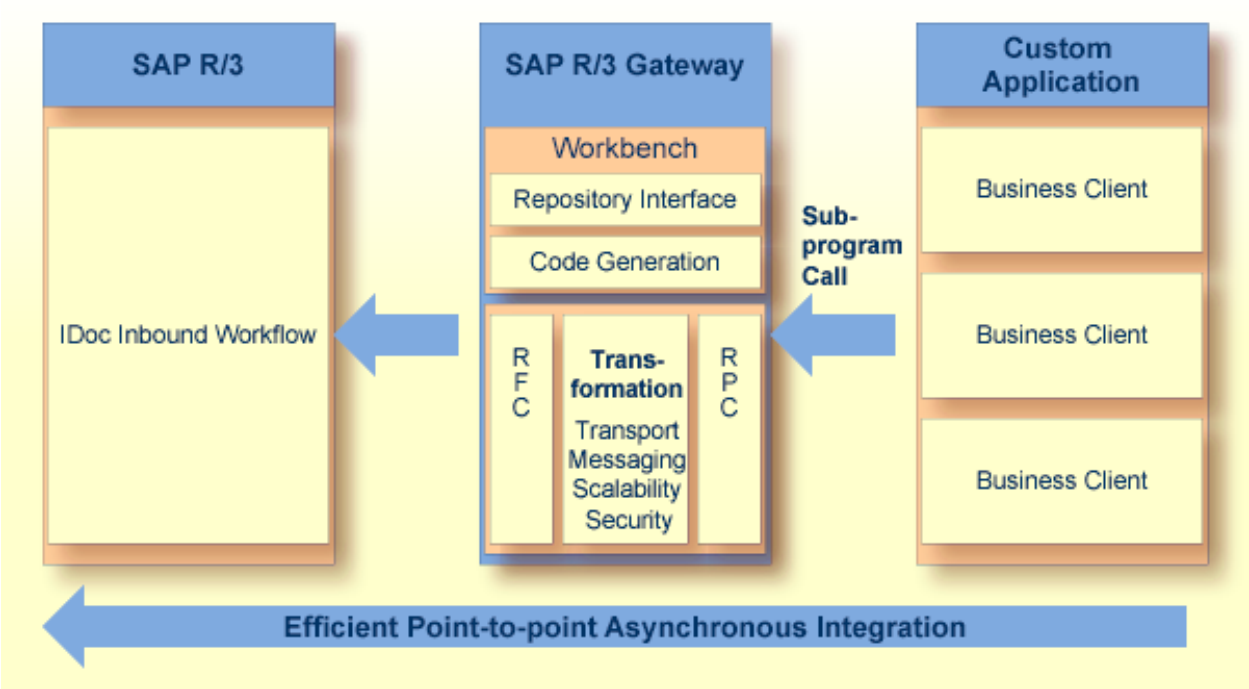
This scenario makes it possible to call your existing functions from SAP R/3. This synchronous method reuses the business interfaces, for example, if you already have complex evaluation algorithms implemented that should send calls from your own SAP R/3 application.



Use the development kit to implement this type of integration. See the section **Develop an SAP Call to an External Application (Rfc2Rpc Kernel)**.

Custom Application sends IDoc

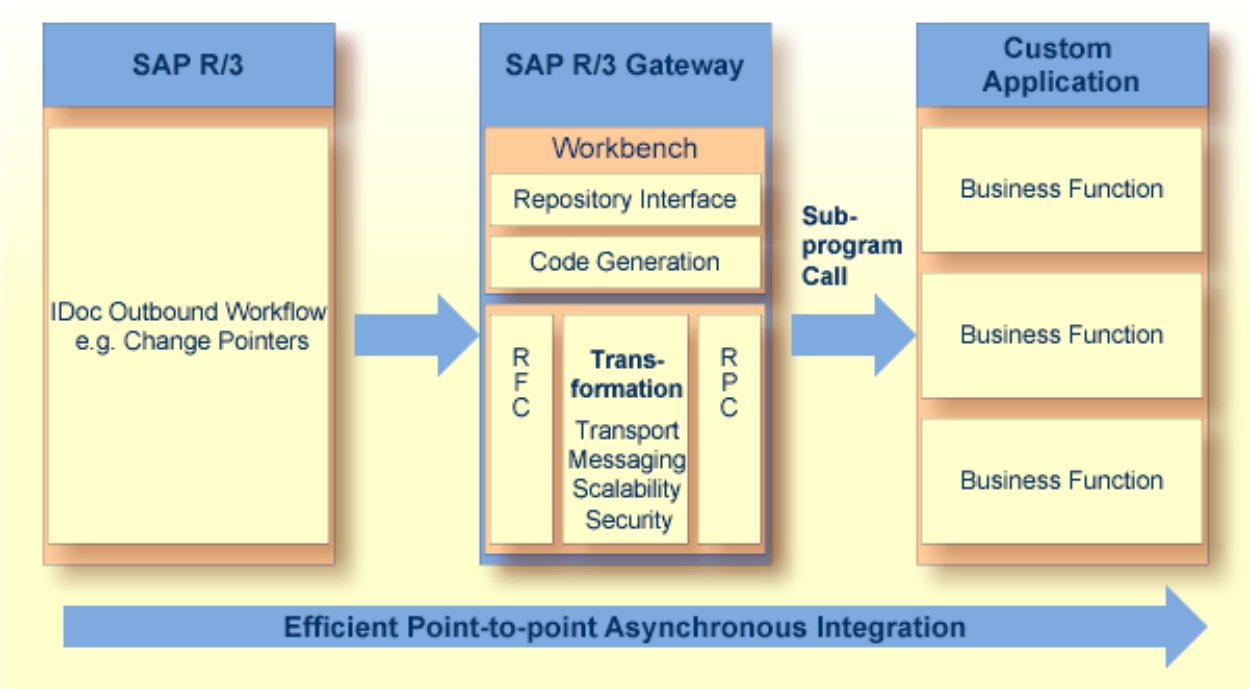
SAP R/3 delivers several document types for asynchronous exchange.



This scenario supports the updating of mass data in SAP R/3. The **IDoc XML Gateway** helps you to implement the interface provided by IDoc to your business application.

SAP delivers IDoc to Customer Application

To notify your business application of a change to mass data, your application can receive IDoc documents. The business application will be notified by a subprogram call and receives the data with your business interface.



The **IDoc XML Gateway** helps you to implement this scenario.

5 Installation and Configuration

This chapter tells you what you must know and do to install and use the SAP R/3 Gateway successfully. Before performing the installation, please read the Software AG Legal Notice.

Prerequisites

The following are the prerequisites for deploying and running SAP R/3 Gateway. You can also use this information as an installation checklist.

To install the SAP R/3 Gateway

1. Select a gateway machine.
2. Create a user `sag` for SAP R/3 Gateway.
3. Access the gateway machine with Telnet, SSH or Windows Terminal Client (mstsc).
4. Create a `HOME` directory. We recommend creating a directory on a partition with 1 GB of available disk space.
5. Connect the gateway machine with FTP and newly-created user, if no CD-ROM is available for the installation.
6. Ensure that an HTTP connection is available from your local web browser to the gateway machine. This is necessary if your environment runs with a secured proxy or has a Firewall.
7. Ensure that the SAP R/3 Application Server can be reached from your gateway machine.
8. Ensure that you have an ANSI C-Compiler for generating the communication kernels.
9. Under Windows, use the Microsoft Developer Studio C/C++ Compiler version 6 (or higher). The Professional version is sufficient.

   Under UNIX, you will find the GNU C-Compiler for most platforms under **http://gcc.gnu.org**. Alternatively, you can use a compiler provided by your platform vendor.
10. Install webMethods EntireX SDK on the same machine on which the SAP R/3 Gateway is installed, and ensure you have access to the webMethods EntireX Broker. See the section ***Installing webMethods EntireX Components***
11. The SAP RfcSdk is needed for your production environment. You can download this software from SAP's Support System OSS. After downloading, unpack the CAR file with `car -xvf rfc.car` to a temporary location in the file system. Later, after installing SAP R/3 Gateway, move the temporary files to their proper location as described in the section ***Directory Structure***.
12. For your Java environment, JDK 1.3 or higher is required.
13. A web application server with servlet container is required to run the SAP R/3 Gateway Portal. For the current version, deployment has been tested with Tomcat. For more information, see ***Installing and Starting the Web Server***.

14 To process XML documents, you need an XML parser and an XSLT processor. SAP R/3 Gateway uses Xalan and Xerces. For more information, see ***Installing Xerces and Xalan***.

15 When configuring the communication kernels, various connection parameters are required which differ depending on the direction of the communication.

The Rpc2Rfc kernel needs:

- CPIC User ID
- Password
- Client
- System number
- DNS name of Application Server

The Rfc2Rpc kernel needs:

- DNS name of Application Server
- TCP/IP Port of Application Server
- Program ID of RFC-Destination created by **SM59**

You should know the value of these parameters before starting work with SAP R/3 Gateway.

Installing webMethods EntireX Components

Install webMethods EntireX SDK on your gateway machine platform. Under UNIX, create a `sag` user. Install the RPCServer from the webMethods EntireX SDK on the gateway machine (see component list below).

The installation of webMethods EntireX Broker is optional if you already have an installation on another platform. In this case, you must modify the webMethods EntireX Broker Attribute File to establish connectivity.

During the installation process, you are prompted to select from the following components (required components are indicated):

- webMethods EntireX Common Files (please select).
- webMethods EntireX Broker (select if not already installed on another platform).
- webMethods EntireX Runtime:
  - DCOM Runtime (optional, not required).
  - Developer's Kit Runtime (please select).
- webMethods EntireX SDK:

  - DCOM SDK (optional, not required).
  - Developer's Kit SDK (please select).
- webMethods EntireX Starter Kit (optional, not required).
- System Management Hub (optional, not required).
- Extended Transport Service (please select).
- Software AG Common Tomcat (please select).

Installing and Starting the Web Server

The SAP R/3 Gateway is delivered as a web application. For this reason you need a web server. Currently, the deployment process has only been tested with **Tomcat 4**. Optionally, it is possible to use an existing **JBoss** as web application server.

Create a User (UNIX)

It is easier for your UNIX Administrator to declare a separate user to the operating system to run the application server because the SAP R/3 Gateway starts the kernels as subprocesses. The user of the application server must have the same group as the `sag` user.

Using JBoss

The SAP R/3 Gateway can run in **JBoss**. This makes sense only if you already have a running JBoss installation with existing applications and you do not want to install a Tomcat, too. JBoss delivers many features which are not used by SAP R/3 Gateway.

The deployment of SAP R/3 Gateway with JBoss differs from deployment with Tomcat, because the JBoss does not unpack the web application.

To deploy SAP R/3 Gateway with JBoss

1 Unpack the *sapr3gateway.war* file into a directory with the name*sapr3gateway.war*. Use the ZIP utility or the command

```
jar -xvf sapr3gateway.war
```

2 Copy the directory *sapr3gateway.war* to JBoss *server\default\deploy*. After copying, create a subdirectory *sapr3gateway.war* in this directory.

3 (Re-) start JBoss.

4 Go to **License Agreement** and **Setup Wizard**.

5 Check **Xalan installation**.

Using Tomcat 4

Download a version from ***http://tomcat.apache.org/index.html***.

The Windows installation of Jakarta Tomcat supports configuration as a Windows Service. We recommend installing Tomcat without spaces in the directory names. You can avoid this restriction by using the short directory name or later in the **Setup Wizard**. To evaluate the short directory name, use the DOS command `dir /x`. For example, replace the long name of the `c:\Program Files\...` installation directory by `c:\PROGRA~1\...` during the Tomcat installation setup.

Under UNIX after installation of SAP R/3 Gateway, you will find a script to start Tomcat at boot time. This is described in the section **Startup at Boot Time**, below.

Using Tomcat 5, 5.5 or Software AG Common Tomcat

The SAP R/3 Gateway can run inside Tomcat 5. From **Tomcat 4** to 5, the mail API is dropped. To solve this problem, please copy *mail.jar* and *activation.jar* into *$TOMCAT_HOME/common/lib* directory.

Update 11 contains the mail API inside the SAP R/3 Gateway web application. There are no additional installation steps for the web application server.

Configure Tomcat Users

To control the web deployment in Tomcat, one user must be created with the roles of "manager" and "admin". Create and edit a user in the *tomcat/conf/tomcat-users.xml* file.

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager"/>
  <role rolename="admin"/>
  <role rolename="sapr3gateway"/>
  <user username="admin" password="admin" fullName="" roles="admin,manager"/>
</tomcat-users>
```

Add a role "sapr3gateway" if you want to secure the SAP R/3 Gateway application itself. Later, you will see that creating an Access Control List depends on the logged-in users.

Startup at Boot Time

Start the application server when the operating system is booted. There is a script example in the file *webapps/sapr3gateway/setup/tomcat.sh*.

```
#!/bin/sh -ex
#

JAVA_HOME=/usr/j2se
export JAVA_HOME

TOMCAT_HOME=/home/sag/jakarta-tomcat-4.1.31
export TOMCAT_HOME

# Set memory requirements for IDoc XML Gateway
#   -Xms for initialization
#   -Xmx for maximum
JAVA_OPTS="-Xms100M -Xmx500M"
export JAVA_OPTS

# Run Tomcat under control of ...
userid=sag

case "$1" in
start)
  su $userid -c "rm $TOMCAT_HOME/logs/catalina.out"
  su $userid -c "nohup $TOMCAT_HOME/bin/startup.sh >$TOMCAT_HOME/logs/nohup.log"
  ;;
restart)
  su $userid -c "$TOMCAT_HOME/bin/shutdown.sh"
  su $userid -c "rm $TOMCAT_HOME/logs/catalina.out"
  su $userid -c "nohup $TOMCAT_HOME/bin/startup.sh >$TOMCAT_HOME/logs/nohup.log"
  ;;
stop)
  su $userid -c "$TOMCAT_HOME/bin/shutdown.sh"
  ;;
*)
echo "Usage: $0 {start|stop|restart}"
exit 1
;;
esac

exit 0
```

Adapt and copy this file to */etc/init.d* (on Sun Solaris or Linux) and create soft links:

```
ln -s /etc/init.d/tomcat /etc/init.d/rc3.d/S90tomcat
ln -s /etc/init.d/tomcat /etc/init.d/rc3.d/K10tomcat
```

Note: On Linux, there is already a tomcat startup script in */etc/init.d*. Adapt the existing one or use this one.

Installing Xerces and Xalan

Xerces (***http://xerces.apache.org/xerces2-j/index.html***) is used to read and write XML documents and Xalan (***http://xml.apache.org/xalan-j/index.html***) is used to generate output for the Browser application.

You can download Xalan from the web site ***http://xml.apache.org/xalan-j/index.html*** or use the delivered version in the *3rdparty/xalan* directory on CD. The following table contains information on installation.

Component	JDK 1.3	JDK 1.4
Xerces	Is included and running in web application. See ***Installing the Gateway Portal***.	Is included.
Xalan	Is included and running in web application. See ***Installing the Gateway Portal***.	An older version is included in JVM or Tomcat. Therefore, use a version 2.4 or 2.5 (not 2.6.0) and install in Tomcat's directory *common/endorsed*. Another possibility is to use the Endorsed Standards Override Mechanism. Place the *xalan.jar*, in the `<java-home>\jre\lib\endorse`d directory, where `<java-home>` is where your JDK is installed. With JBoss, you must use the last described mechanism.

After installing Xerces or Xalan into Tomcat or JDK, you must restart the web server (see the section **Installing and Starting the Web Server**).

Tip: After ***Installing the Gateway Portal***, it is possible to check the running Xerces and Xalan versions by clicking **Help** and choosing **Version Info** (*http://YourGateway:8080/sapr3gateway/manager/version*) and Xalan Version (*http://YourGateway:8080/sapr3gateway/manager/xalanVersion*)

Directory Structure

The following table illustrates the directory structure on your UNIX file system after all of the required components have been installed:

Directory	Description
$HOME/sag	General home directory
$HOME/sag/exx	webMethods EntireX installation
$HOME/sag/exx/vXXX	webMethods EntireX Installation, version XXX
$HOME/sag/tomcat	Application Server installation
$HOME/sag/tomcat/webapps/sapr3gateway	SAP R/3 Gateway home directory
$HOME/sag/tomcat/webapps/sapr3gateway/RfcSdk.XXX	Installation of SAP RfcSdk. XXX is the name of the platform
$HOME/sag/tomcat/webapps/sapr3gateway/WEB-INF/lib	Installed Open Source components Xerces and Xalan
$HOME/sag/tomcat/common/endorsed	Installed Open Source components Xerces and Xalan for all applications

License Agreement

Before you complete the next installation commands, you must read and accept the terms of Software AG's Legal Notice.

Installing the Gateway Portal

SAP R/3 Gateway is delivered as a web application. Install this web application using one of the following steps:

- The installation CD contains one *sapr3gateway.war* file for all Windows platforms and one for all UNIX target platforms. Select one of these from one of the subdirectories *windows* or *unix*.
- Copy *sapr3gateway.war* to the *tomcat/webapps* directory, or
- Use Tomcat Manager (or *http://YourGateway:8080/manager/html/list*) to upload *sapr3gateway.war* from the local file system. The application server will transport the package and automatically unpack it.
- If you use JBoss as web application server, install the Gateway Portal as described in the section **Using JBoss**.

Once the web application has been installed, you can access SAP R/3 Gateway using the following URI: *http://YourGateway:8080/sapr3gateway/manager/index*.

Optionally, it is possible to install the IDoc XML Gateway with *sapr3idocxmlgateway.war* with the same steps that are described above. More installation hints are described in the section **Installation**.

Using the Setup Wizard

After you have deployed the gateway portal, you must start the Setup Wizard by clicking **Help** and choosing **Setup Wizard**; or from *http://YourGateway:8080/sapr3gateway/manager/setupWizard*. This wizard evaluates your environment and prompts you for more information.

- Activate application security.
- Choose the platform-dependent makefile.
- Set the path to the SAP RfcSdk.
- Set the path to C-Compiler.
- Set SAG home, SAG environment script or environment variable to Windows Operating System.

Please read the **Release Notes** for more installation and configuration information about related products.

Optionally, it is possible to start the **setup wizard** for the IDoc XML Gateway.

III Programming and Running SAP R/3 Gateway

The SAP R/3 Gateway provides a development environment and a running task environment. The development environment is used to generate communication code, and the running task environment to define the runtime parameters.

This part of theSAP R/3 Gateway documentation describes the activities you must perform in both environments. This information is provided under the following headings:

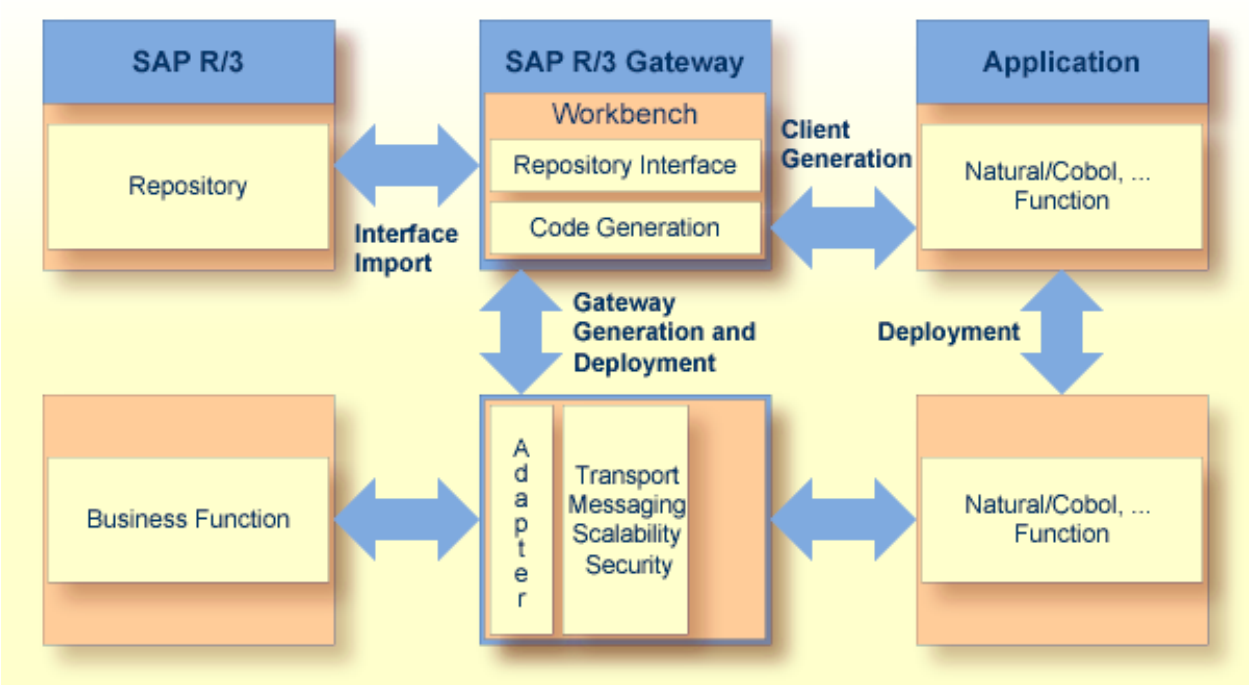
6 Overview of Development

After analyzing your business process, develop the interfaces for your communication process. Depending on the events in your client application, you must describe the interface in an webMethods EntireX Interface Definition Language (IDL).

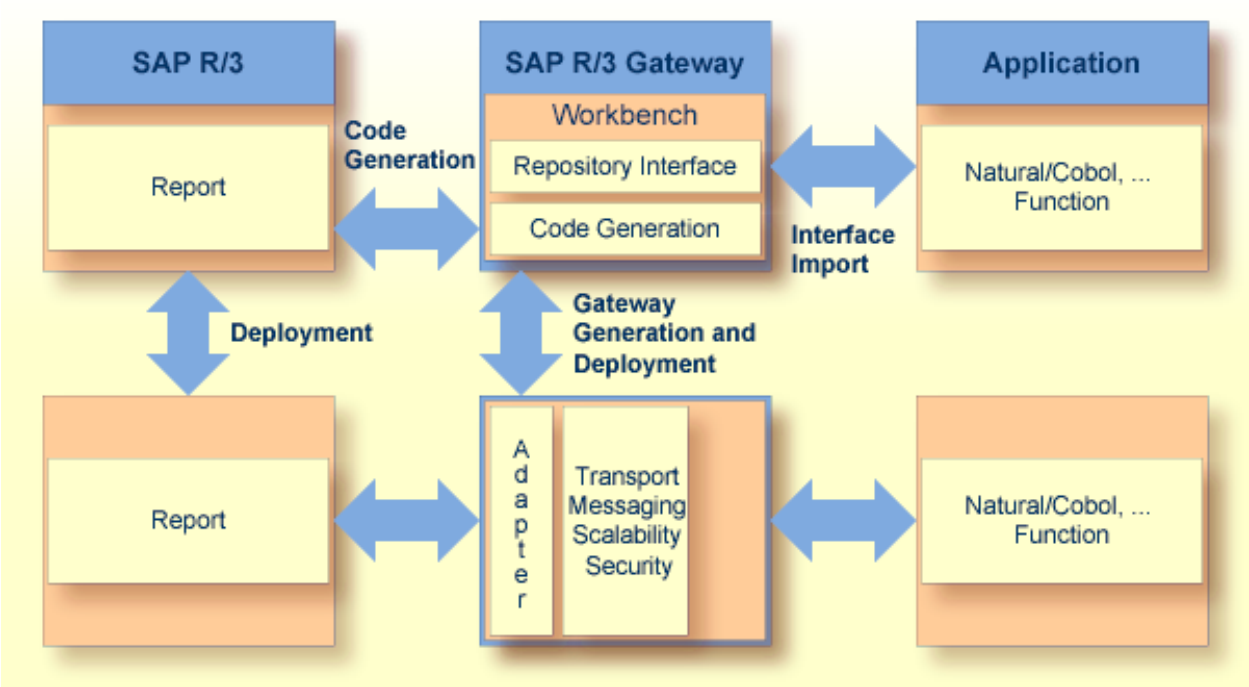
Develop a client call to SAP R/3

Develop an SAP call to an external application

The following picture illustrates the development process and the components required for a communication event created in your business application.



The following picture illustrates the development process and the components required for a communication event created in SAP R/3.



7 Develop a Client Call to SAP/R3 (Rpc2Rfc Kernel)

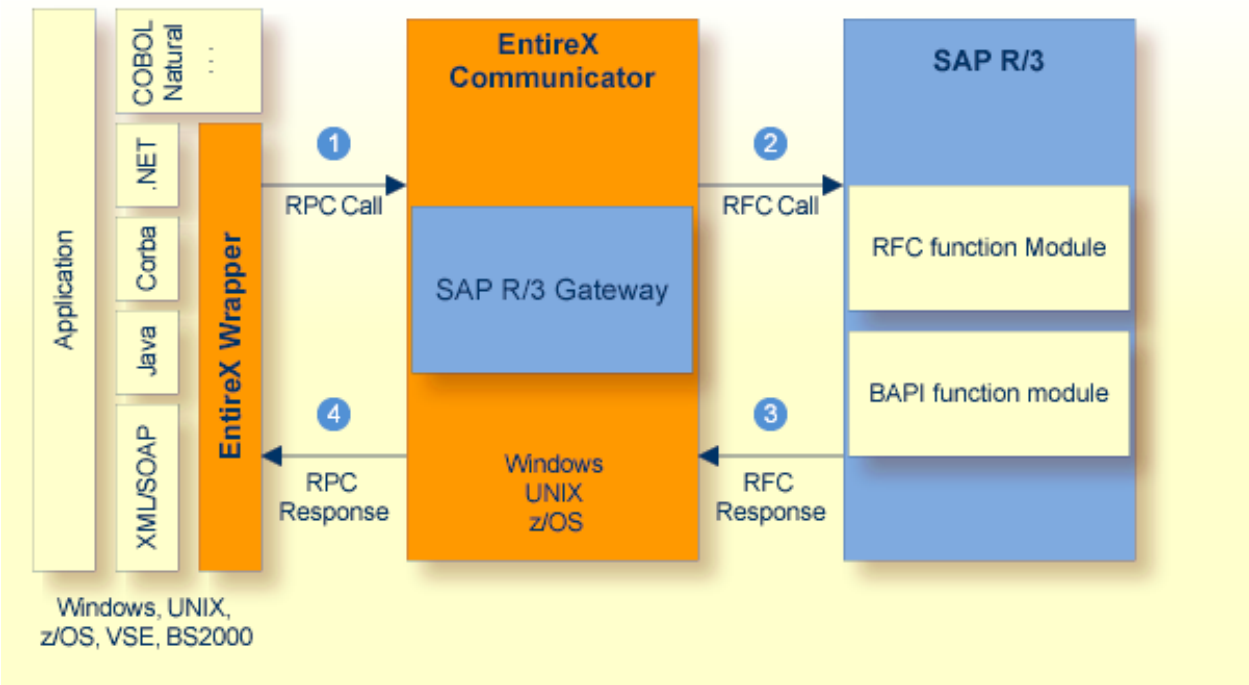
This chapter describes how to develop an webMethods EntireX RPC client which can call an SAP R/3 ABAP business function. The following topics are covered here:

Overview

The following picture shows the technical communication (send/response) for an integration scenario involving a client call to SAP.

The communication runs as follows:

1. Your application creates an RPC call to webMethods EntireX Broker.
2. The SAP R/3 Gateway receives this call and transforms it to an RFC call to the SAP application server.
3. The ABAP (RFC enabled) function is called and then replies with an RFC to webMethods EntireX Broker.
4. The SAP R/3 Gateway receives the reply and transforms it to an RPC call to the calling program.



To develop this scenario

1 Define the ABAP function and create the IDL (**Interface Definition Language**).

2 Paste the new IDL into the existing IDL.

3 Compile and link the IDL to generate the executable kernels.

4 Download the IDL to the client environment.

5 Write the first client.

These steps are described in detail in the following sections.

Generate IDL

First, you must define for your client application which ABAP function you want to call. You can do this and generate the IDL either by using the browser page in the SAP R/3 Gateway portal, or from the command line. This section describes both methods as well as some additional programming hints under the following headings:

- Prerequisites
- Generating IDL using the Portal Text Editor
- Generating IDL using the Portal IDL Editor
- Generating IDL from the Command Line
- Build Different Views to the Same Business Function
- Long Strings
- Multiple Group and Tables
- Long Field Names
- Reserved Field Names

Prerequisites

The following sections describe the process of IDL generation. During this process, the RFC IDL tool is used. This tool acts as an EntireX RPC client and requires a running Rpc2Rfc kernel.

To generate IDL

1 As RPC client, the connection parameter to the Broker and service (CLASS/SERVER/SERVICE) must be set. To check or change the parameter, open the **System Constants** page from the **Configuration** menu. Refer also to the description of the ***System Constants Parameters***.

If the webMethods EntireX Broker does not reside on the `localhost`, you can change the RFC IDL Tool Host Address.

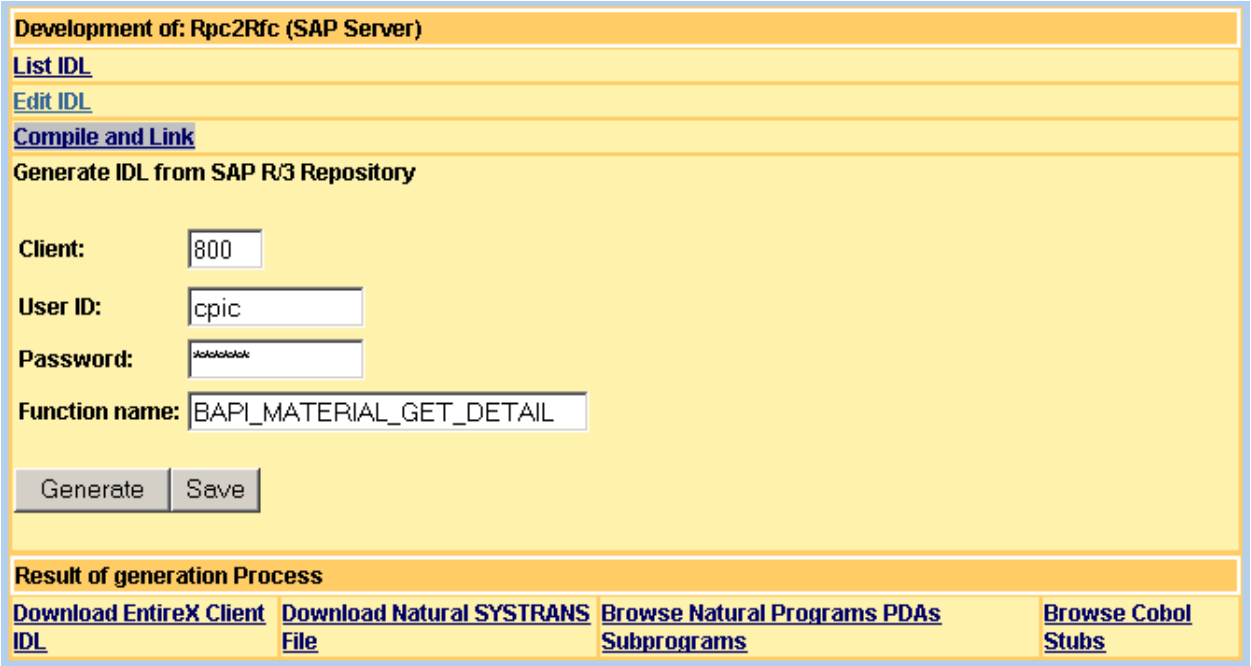
2 **Compile and Link** existing delivered IDL for the Rpc2Rfc kernel.

3 **Deploy** new compiled shared libraries (on UNIX) or DLLs (on Windows).

4 The RPC Server must be running to use the RPF IDL tool. More information about starting this kernel process is provided in the section ***Running Task Rpc2Rfc (Client calls SAP) Kernel Environment***.

Generating IDL using the Portal Text Editor

You can use the **Development** page of the SAP R/3 Gateway portal to generate the IDL.



With List IDL, you can display the existing IDL at the end of a new browser page.

To generate IDL

1 Type your new ABAP function in the field `Function name` and choose **Generate**.

This loads a new page with the IDL from the SAP Repository, as illustrated in the following figure:

Job Output - Return Value: 0

```
Call /FS/fs0225/home/thr/64/sagenv.711
[Setting environment for SAG BTY]
[done]
[Setting environment for System Management Hub 3.2.1.5]
[done]
[Setting environment for Extended Transport System]
[done]
Warning: CSLNK is not set ! Setup WCP if NET-WORK is to be used.
[Setting environment for EntireX XML Mediator]
[done]
Program '' : 'BAPI_MATERIAL_GET_DETAIL' Is
  Define Data Parameter
    1 RFC_SYSTEM  In Out
      2 userid       (A12)
      2 passwd       (A8)
      2 client       (A3)
      2 ok           (L)
      2 message      (A250)
      2 operation    (I2)
      2 handle       (I4)
      2 transaction  (A24)
    1 MATERIAL In /* BAPIMATDET
      2 Material     (A18)                /*
    1 PLANT In /* BAPIMATALL
      2 Plant        (A4)                 /*
    1 VALUATIONAREA In /* BAPIMATALL
      2 ValArea      (A4)                 /*
    1 VALUATIONTYPE In /* BAPIMATALL
      2 ValType      (A10)                /*
    1 MATERIALPLANTDATA Out /* BAPIMATDOC
      2 PurGroup     (A3)                 /* Purchasing group
      2 IssueUnit    (A3)                 /* Unit of issue
    1 MATERIALVALUATIONDATA Out /* BAPIMATDOBEW
      2 PriceCtrl    (A1)                 /* Price control indicator
      2 MovingPr     (P12.4)              /* Moving average price/periodic unit price
      2 StdPrice     (P12.4)              /* Standard price
      2 PriceUnit    (P3.0)               /* Price unit
      2 Currency     (A5)                 /* Currency Key
      2 CurrencyIso  (A3)                 /* ISO code currency
    1 MATERIAL_GENERAL_DATA Out /* BAPIMATDOA
      2 MatlDesc     (A40)                /* Material description
      2 OldMatNo     (A18)                /* Old material number
      2 MatlType     (A4)                 /* Material type
      2 IndSector    (A1)                 /* Industry sector
      2 Division     (A2)                 /* Division
      2 MatlGroup    (A9)                 /* Material group
      2 ProdHier     (A18)                /* Product hierarchy
      2 BasicMatl    (A14)                /* Basic material (basic constituent of a material) - obsolete
      2 StdDescr     (A18)                /* Industry Standard Description (such as ANSI or ISO)
      2 LabDesign    (A3)                 /* Laboratory/design office
      2 ProdMemo     (A18)                /* Production/inspection memo
      2 Pageformat   (A4)                 /* Page Format of Production Memo
      2 Container    (A2)                 /* Container requirements
      2 StorConds    (A2)                 /* Storage conditions
      2 TempConds    (A2)                 /* Temperature conditions indicator
      2 BaseUom      (A3)                 /* Base unit of measure
      2 EanUpc       (A18)                /* International Article Number (EAN/UPC)
      2 EanCat       (A2)                 /* Category of International Article Number (EAN)
```

2 Copy the generated contents to your clipboard and choose **Edit IDL**.

This starts a text editor at the end of the new page.

Paste the contents of the clipboard to the end of the existing IDL.

Edit IDL for: Rpc2Rfc (SAP Server)

```
Program '' : 'BAPI_MATERIAL_GET_DETAIL' Is
  Define Data Parameter
    1 RFC_SYSTEM  In Out
      2 userid       (A12)
      2 passwd       (A8)
      2 client       (A3)
      2 ok           (L)
      2 message      (A250)
      2 operation    (I2)
      2 handle       (I4)
      2 transaction  (A24)
    1 MATERIAL In /* BAPIMATDET
      2 Material        (A18)           /*
    1 PLANT In /* BAPIMATALL
      2 Plant   (A4)            /*
    1 VALUATIONAREA In /* BAPIMATALL
      2 ValArea         (A4)            /*
    1 VALUATIONTYPE In /* BAPIMATALL
      2 ValType         (A10)           /*
    1 MATERIALPLANTDATA Out /* BAPIMATDOC
      2 PurGroup        (A3)            /* Einkäufergruppe
      2 IssueUnit       (A3)            /* Ausgabemengeneinheit
    1 MATERIALVALUATIONDATA Out /* BAPIMATDOBEW
      2 PriceCtrl       (A1)            /* Preissteuerungskennzeichen
      2 MovingPr        (P12.4)                 /* Gleitender
Durchschnittspreis/Periodischer Verrechnungspreis
      2 StdPrice        (P12.4)                 /* Standardpreis
      2 PriceUnit       (P3.0)          /* Preiseinheit
      2 Currency        (A5)            /* Währungsschlüssel
      2 CurrencyIso     (A3)            /* ISO Code Währung
```

Save

3 Define an 8-character-long function name between the ' ' characters after the string "Program". The long alias name is used by the ABAP server. Later, in your COBOL or Natural business application, the ABAP function is called using the short name you define here. With this step, you can reduce a 32-character function name to 8 characters.

The following example shows a short name MATGETDT for the long ABAP BAPI function name BAPI_MATERIAL_GET_DETAIL.

Edit IDL for: Rpc2Rfc (SAP Server)

```
Program 'MATGETDT' : 'BAPI_MATERIAL_GET_DETAIL' Is
  Define Data Parameter
   1 RFC_SYSTEM        In Out
    2 userid        (A12)
    2 passwd        (A8)
```

4 On the **Development** page, type the additional parameters for the RFC IDL generation tool:

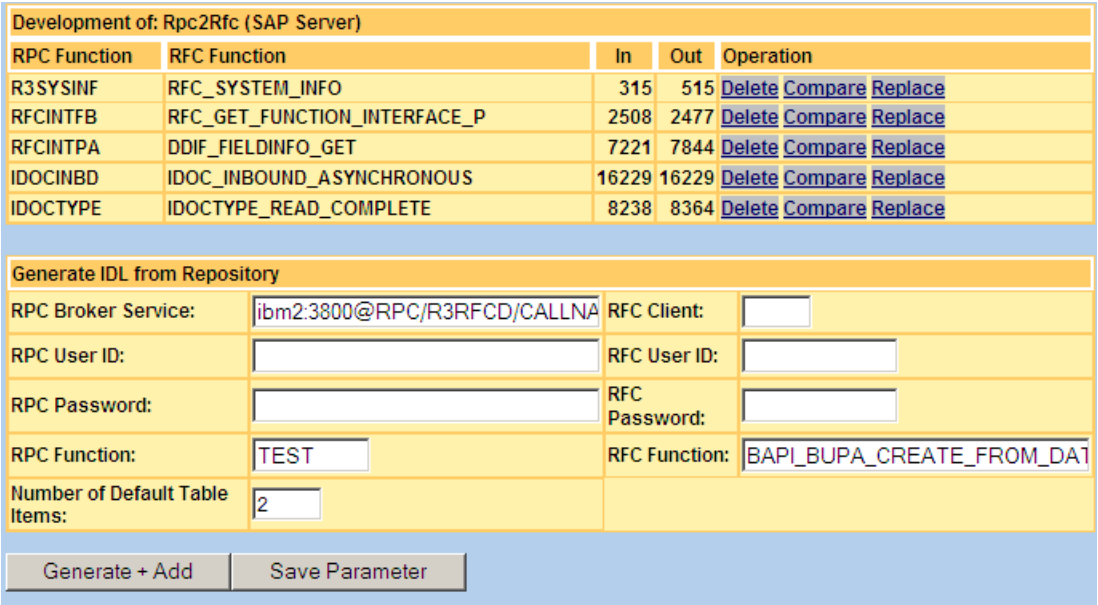
Client	The 3-character client ID definition.
User ID	Please ask your SAP administrator for an RFC (Batch/CPIC) user definition.
Password	RFC password to go with the user ID.

These parameters are passed to the running **Rpc2Rfc kernel**. Depending on the parameter Gateway Logon Strategy (EXX_GW_LOGON_PRIO), the Rpc2Rfc kernel will evaluate the logon.

5 Choose **Save** to save the above parameters and the ABAP function name. The saved parameters are now persistent, that is, when you load this page later, the same values will be displayed.

Generating IDL using the Portal IDL Editor

The IDL editor is accessible from the development page. First, all included functions are listed with their short and long name. To modify the IDL, use the available commands.



The `In` and `Out` column shows the size of the transfer buffer for each request. This can help to optimize the communication between client and server. The following table defines the parameter and the available commands.

Command	Description
Generate + Add	Generates the IDL of a given function in the field from the SAP repository and adds it at the end of the existing IDL file.
Save	Saves all parameters in the configuration of this development environment.
Delete	Deletes the selected function from the IDL and saves the new IDL file. The old IDL file is backed up in the history directory.

Command	Description
Compare	Compares the selected function in the IDL file with the respository contents; differences between the two files will be displayed on the next page. Connection parameters (Broker, service, user IDs and passwords) are reused from the `Generate IDL from Repository` dialog. The comparison algorithm tries to find the locally defined groups and fields in the repository. You will be notified only in cases where the number of groups and/or fields differ. If there are more fields defined in a local function than in the repository, a detailed description of the missing fields will be displayed. The following picture shows the result of a comparison between the delivered local `RFC_SYSTEM_INFO` IDL function (generated by SAP R/3 3.1) and the SAP ECC 6.0 repository. The protocol only displays listings when fields or group parameter have been added or renamed. This means that the previous interface definition can call the new one. Result of 'R3SYSINF/RFC_SYSTEM_INFO' Comparison | Local IDL | Repository Group or field parameter is detected at other position. | RFCSI_EXPORT | CURRENT_RESOURCES detected Different number of child fields of parameter R3SYSINF.RFCSI_EXPORT. | 18 fields | 20 fields Different field names detected. | RFCSI_EXPORT.rfcproto | RFCSI_EXPORT.Rfcproto Different field names detected. | RFCSI_EXPORT.rfcchartype | RFCSI_EXPORT.Rfcchartyp During a migration process this comparison report shows the differences between the SAP R/3 versions.
Replace	Replaces the selected function in the IDL by the respository contents. Connection parameters (Broker, service, user IDs and passwords) are reused from the `Generate IDL from Repository` dialog.

Parameter	Description
RPC Broker Service	EntireX Broker and service address of Rpc2Rfc server.
RPC User ID	EntireX user ID for logon with security.
RPC Password	EntireX password for logon with security.
RPC Function	Sets 8-character function name in IDL. This function name should not exist in IDL.
RFC Client	R/3 logon with 3-character client name.
RFC User ID	R/3 logon user ID.
RFC Password	R/3 logon password for user ID.
RFC Function	Retrieves the interface description from this R/3 function.
Number of Default Table Items	Sets the number for the array size of tables (multiple groups) in the IDL.

Generating IDL from the Command Line

It is possible to generate the IDL from the SAP Repository using the command line tool RfcIdl. It is the same tool used by the browser page.

The command line utility must have the mandatory parameters for calling:

- webMethods EntireX Broker server address
- ABAP function name
- webMethods EntireX short function name

The utility writes the following output if called without parameters:

```
Usage parameter:
 -a<Broker ServerAddress>
 -e<Function short name>
 -f<SAP R/3 function name>
 -u<SAP R/3 user id>
 -p<SAP R/3 password>
 -c<SAP R/3 client>
 -o<ABAP function tablename>=<occurance>
 -x<ABAP function excluded tablename>
 -r<EntireX user id>
 -d<EntireX password>
 -t<Number of default table items>
```

The following example gets the IDL from the ABAP function BAPI_MATERIAL_GET_DETAIL:

```
rfcidl -alocalhost:1971@RPC/R3RFCD/CALLNAT -eMATGETDL -fBAPI_MATERIAL_GET_DETAIL
```

The IDL is then printed on the standard output:

```
Program 'MATGETDL' : 'BAPI_MATERIAL_GET_DETAIL' Is
  Define Data Parameter
    1 RFC_SYSTEM  In Out
      2 userid       (A12)
      2 passwd       (A8)
      2 client       (A3)
      2 ok           (L)
      2 message      (A250)
      2 operation    (I2)
      2 handle       (I4)
      2 transaction  (A24)
    1 MATERIAL In /* BAPIMATDET
      2 Material  (A18)   /* Materialnummer
    1 MATERIAL_GENERAL_DATA Out /* BAPIMATDOA
      2 MatlDesc  (A40)   /* Materialkurztext
      2 OldMatNo  (A18)   /* Alte Materialnummer
```

```
      2 MatlType  (A4)   /* Materialart
      2 IndSector  (A1)   /* Branche
```

Build Different Views to the Same Business Function

Normally, each generated SAP R/3 ABAP business function is inserted in the IDL as one program. It is possible, however, to create more than one program in the IDL to call the same ABAP function.

For example:

```
Program 'FU_1'Is
   Define Data Parameter
    1 BAPI_MATERIAL_GETLIST (A1)
    1 RFC_SYSTEM        In Out
     2 userid        (A12)
...
   End-Define

Program 'FU_2'Is
   Define Data Parameter
    1 BAPI_MATERIAL_GETLIST (A1)
    1 RFC_SYSTEM        In Out
     2 userid        (A12)
```

In this case, no alias is defined in the IDL. Instead, the gateway gets the ABAP function name from the first parameter. You can now write different parameter groups for programs FU_1 and FU_2.

Long Strings

Assume, by way of example, the ABAP function requires a string with a length of 1000 characters. Your client (Natural) cannot allocate a string of this length. The solution is to change the field to a multiple field.

For example:

```
...
    1 MATERIAL_GENERAL_DATA Out
      2 MatlDesc  (A1000)
...
```

Change the field MatlDesc to a multiple field. The allocated memory must be the same.

```
...
    1 MATERIAL_GENERAL_DATA Out
      2 MatlDesc  (A200/1:5)
...
```

Multiple Group and Tables

The default IDL generation prints 10 occurrences for each table. The section ***Table Count, Offset and Fill Parameter*** describes how to handle more occurrences. You can, of course, define your own number of tables in the IDL.

Long Field Names

ABAP and the webMethods EntireX IDL are restricted to 32-character field names. Each table parameter also requires 3 parameters (count, offset and fill) as postfix (see the section **Table Count, Offset and Fill Parameter**). If the table name has more than 25 characters, you will get an error from the IDL compiler during **Compile and Link** because the limit is reached.

Tip: Cut the field names of table parameters, but not the table name itself.

Generated example:

```
...
    1 SALESORGANISATIONSELECTION_Count  (I4) Out
    1 SALESORGANISATIONSELECTION_Offset (I4) In
    1 SALESORGANISATIONSELECTION_Fill   (I4) In Out
    1 SALESORGANISATIONSELECTION (/1:10) In Out /* BAPIMATRASO
...
```

After cutting the table parameters:

```
...
    1 SALESORGANISATIONSEL_Count  (I4) Out
    1 SALESORGANISATIONSEL_Offset (I4) In
    1 SALESORGANISATIONSEL_Fill   (I4) In Out
    1 SALESORGANISATIONSELECTION (/1:10) In Out /* BAPIMATRASO
...
```

Reserved Field Names

Some field names are reserved for the **Interface Definition Language for webMethods EntireX RPC**. Therefore, these field names are replaced during **Compile and Link**. The following table shows the replacement.

Original field name	Replace with
Program	Prog
Library	Libr
Parameter	Param
Structure	Structure1
Version	Vers

You can change the replacement with the "makefile" function on the **Development** page of the **Configuration** menu (*http://YourGateway:8080/sapr3gateway/manager/devAdmin?cfg=1*).

History of the IDL File

During each save step of the IDL file, it is copied to a separate subdirectory with a current time stamp. The **Development** page of the **Configuration** menu (*http://YourGateway:8080/sapr3gateway/manager/devAdmin?cfg=1*) contains the backup template with the specific sub-directory information.

Compile and Link the IDL

Click the **Compile and Link** button on the Development page to start the makefile. The kernels for the executables are generated from the IDL. On the resulting page, you will see the job output of this process. Please check that the job output reports response code 0.

To activate the newly generated version for the runtime environment, you must deploy the executables (see the section **Deploy Kernels**) and restart the Running Task (see ***Running Task Rpc2Rfc (Client calls SAP) Kernel Environment***).

Download webMethods EntireX Client IDL

You can download the client IDL using the button **Download EntireX Client IDL** on the development page. This IDL file can be used in the webMethods EntireX Workbench to generate DCOM, Java or XML clients.

Download Stub for Natural Client

For each defined ABAP function, the compile and link process generates in Natural

- a Parameter Data Area (PDA),
- a subprogram for stub generation,
- an example test program.

Everything is included in one SYSTRANS file, which you can download and import to your Natural Development Environment. On the first SYSTRANS upload call, you must stow the general Natural modules RFCSYSTM and RFCSYS-L.

Choose **Download Natural SYSTRANS File** to save the generated SYSTRANS file to a local disk.

The RFCSYSTM LDA contains the technical field definition for each function. The Natural object RFCSYS-L contains constants for the technical field `operation`. See the section *Write the First Client* for more information.

Although it is possible to use the same fieldname on levels 1 and 2, for Natural the fieldnames within one PDA must be unique. You must therefore change the fieldnames on level 2.

Use the SYSTRANS Utility to import the file. The following is an example job for the mainframe environment:

```
//GKTTRANS JOB USER,CLASS=K,MSGCLASS=X,REGION=8M
//* ------------------------------------------------------------------
//*   SYSTRANS
//* ------------------------------------------------------------------
//NATLOAD  EXEC NATURAL
//CMWKF01   DD DISP=SHR,DSN=MYHOME.R3RFC.TEXT
//CMWKF04   DD DISP=SHR,DSN=MYHOME.R3RFC.LIST
//CMSYNIN   DD DISP=SHR,DSN=MYHOME.SAG.LOGIN(LOGINDEF)  --> LOGON SYSUNLD
//          DD *
TRANSCMD LOAD NAT-OBJECT LIBRARY R3RFC NAME *
FIN
/*
//
```

By default, the SYSTRANS file is generated with the decimal character period (.) for the numeric/packed data type. You can **change this by setting** the parameter `DC`

```
$(ERXIDL) -D OUT_DIR=$(PDA_DIR) -D "DC=," -t nat_trans.tpl
```

to the character in the makefile.

To change the parameter in the command line, go to **Development**, **Development**, **Rpc2Rfc Server ...** and **Makefike** menu item. An editor shows the makefile.

You can also change the destination Natural library name:

```
ERX_LIB_FOR_SYSTRANS=$(ERX_LIB)
```

If your Natural client environment works without stubs (or is stubless), the following parameter `STUBS=OFF` suppresses the generation of subprograms, for example:

```
$(ERXIDL) -D OUT_DIR=$(PDA_DIR) -D STUBS=OFF -t nat_trans.tpl
```

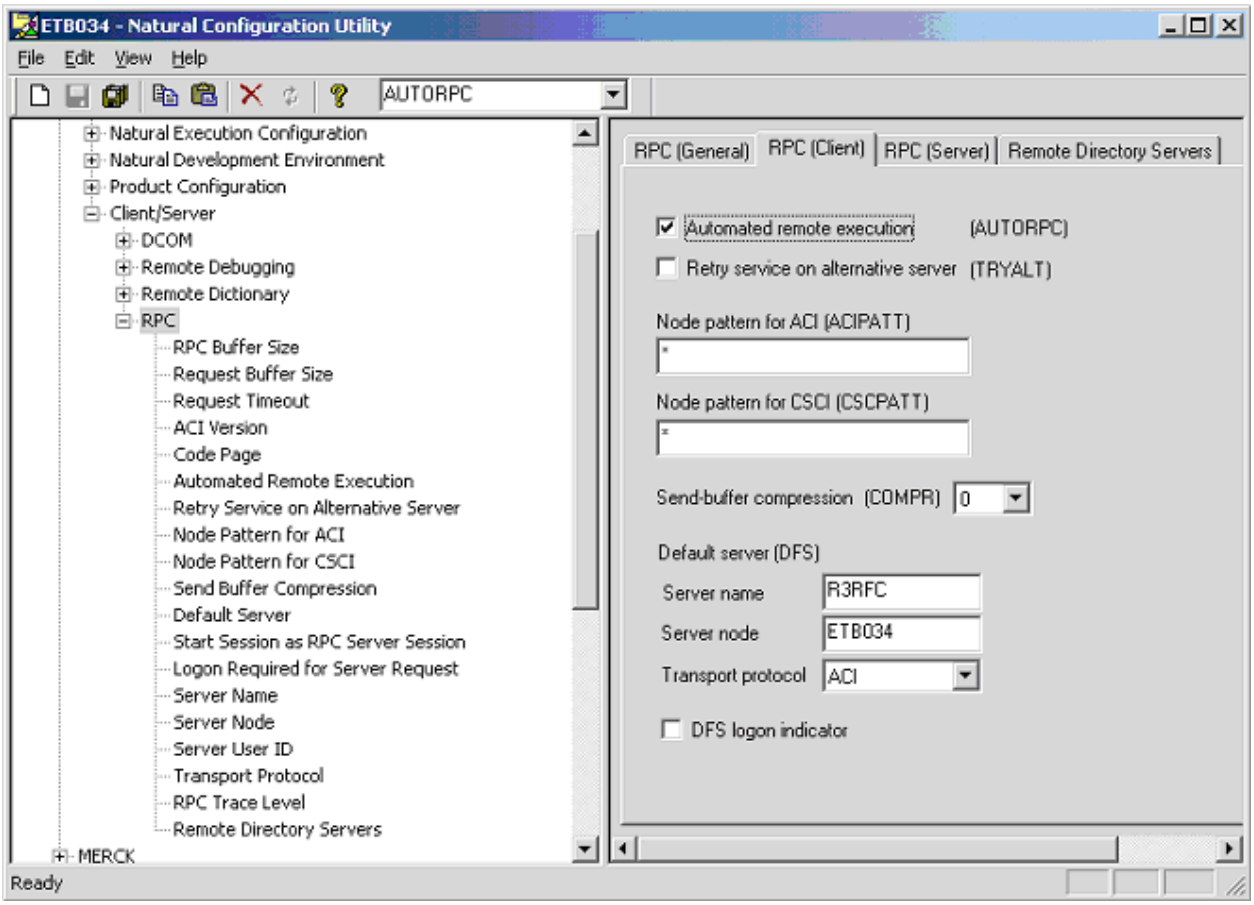
You can use either **Browse Natural Programs PDAs** or **Subprograms** on the development page to look at a Natural program for an example call to an APAB business function. If you have to use stubs in Natural RPC, then you can use the subprograms to build a stub with `SYSRPC`. In other cases, you will find the business interface to the ABAP function in the PDAs.

Configuring the Natural Client

When the SYSTRANS file is first **downloaded** and uploaded to your Natural development environment, the Natural Parameter Module must be adapted as an RPC Client. The tool for setting the parameter depends on the platform.

Caution: Set the compression parameter to: `COMPR=0`. You will receive runtime errors if you set it to `COMPR=1`. This also applies when you use stubs.

The following example shows the Natural Configuration Utility with parameters for a stubless RPC client. The `autorpc` parameter is switched on.



On the mainframe, you can start the Natural nucleus online with the following parameters. In this example, Natural works without stubs.

```
RPC=(SIZE=64,MAXBUFF=32,COMPR=0,AUTO=ON,DFS=(R3RFCD,BKR034, ,ACI))
```

Generate Stub for COBOL Client

Choose **Download COBOL Stubs** on the Development page to browse the generated COBOL source. You can transport the source to your COBOL Development Environment using your file transfer utility (for example, FTP). It is possible to generate an FTP shell script (or batch on Windows) for uploading files.

Please read the **Release Notes** for more EntireX Communicator installation hints.

The COBOL Stubs should be generated during the **Compile and Link** job. This makes it possible to adapt the makefile to your parameters. To set the parameters in System Manager:

1. Go to the **Development** page of the **Configuration** menu or to *http://YourGateway:8080/sapr3gateway/manager/devAdmin*.

2. Select the link to the **Rpc2Rfc (SAP Server)**
3. Select the link to the **COBOL Makefile**. You will get an editor with the makefile.
4. Uncomment the ('#' lines) statements for calling `ERXIDL` and set the parameter `COB_PARMS`. (The back slash character must be the last character of any line that continues onto the next line.) For more information about the parameter, please read the **EntireX Communicator Documentation**.

```
# Define IDL Cobol Template Parameter
#   -D TARGET=BATCH  [ BATCH, CICS ]
#   -D SERVER=SRV1
#   -D BROKER=ETB001
#   -D SECURITY=NONE
#   -D COMM=EXTERNAL [ NONE, EXTERNAL, LINKAGE ]
#   -D QUOTE=0
COB_PARMS= -D TARGET=BATCH  \
           -D SERVER=SRV1   \
           -D BROKER=ETB001 \
           -D SECURITY=NONE \
           -D COMM=EXTERNAL \
           -D QUOTE=0

COBOL: $(IDL_FILE_TMP)
# echo Create Cobol Stubs...
# $(ERXIDL) $(COB_PARMS) -o $(COB_DIR) -t $(EXXTPL)/cobolclient1.tpl ↩
$(IDL_DIR)/$(ERX_LIB)_WithoutAlias.idl
# $(ERXIDL) $(COB_PARMS) -o $(COB_DIR) -t $(EXXTPL)/cobolclient2.tpl ↩
$(IDL_DIR)/$(ERX_LIB)_WithoutAlias.idl
# echo Create Cobol Programs ...
# $(ERXIDL) $(COB_PARMS) -o $(COB_DIR) -t cob_prog.tpl ↩
$(IDL_DIR)/$(ERX_LIB)_WithoutAlias.idl
# echo Create Upload FTP Script ...
# $(ERXIDL) -D SHELL=SH_or_BAT -D LOCAL_DIR=$(COB_DIR) -D REMOTE_ROOT_DIR=.. -t ↩
cob_ftp.tpl $(IDL_DIR)/$(ERX_LIB)_WithoutAlias.idl
# echo Execute FTP Script ...
# chmod +x ftp${ERX_LIB}.sh
# ./ftp${ERX_LIB}.sh host userid password destinationDirectory
```

5. Save the file.
6. Restart the **Compile and Link** job.
7. View the results with the function Browse COBOL Stubs. All COBOL Stubs start with the character `C`. The example programs have the prefix `P`. A program calls its stub with the current parameter data area.

   Tip: The working storage section of a generated program source can be used in your business program.

8. More information about the generated source code is on **IDL COBOL Mapping**.

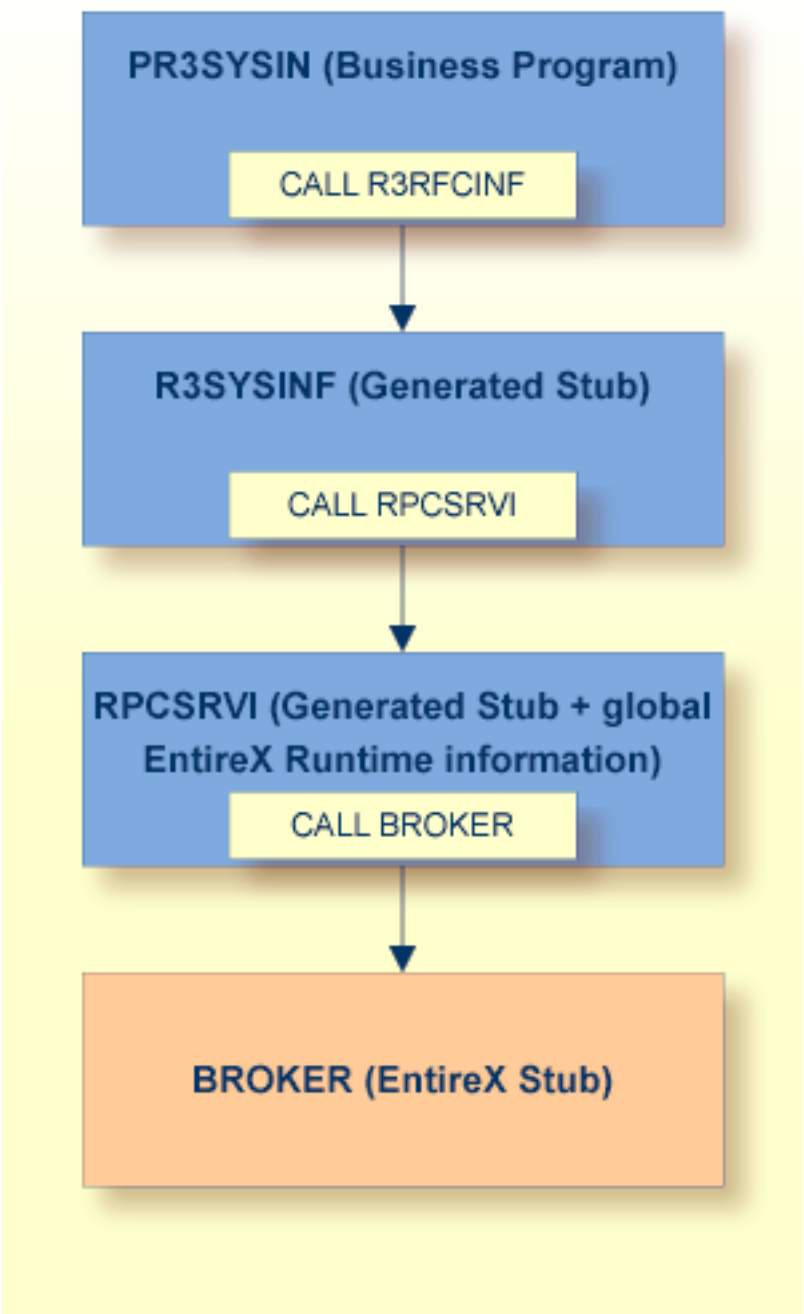
9. Copy the COBOL source with FTP to the mainframe development environment. A shell script is generated with the template `cob_ftp.tpl`. See the example above to call this shell script with the correct FTP parameter. The following listing shows the generated shell script:

```
#!/bin/sh
if [ $# -lt 3 ]; then
  echo usage: $0 host userid password [ destinationDirectory ]
  exit 1
fi

host=$1
userid=$2
passwd=$3
dir=$4
echo open $host                                          > ftp.txt
echo user $userid $passwd                                >>ftp.txt
echo lcd cob                                             >>ftp.txt
if [ $dir ]; then
  echo cd  $dir                                            >>ftp.txt
fi
echo put ERXCOMM.cob   ERXCOMM                           >>ftp.txt
echo put RPCSRVI.cob   RPCSRVI                           >>ftp.txt
echo put CR3SYSINF.cob   R3SYSINF                          >>ftp.txt
echo put PR3SYSINF.cob   PR3SYSIN                         >>ftp.txt
echo dir                                                 >>ftp.txt

ftp -n -i < ftp.txt
```

This UNIX shell script copies all of the necessary COBOL sources to the FTP destination and renames them for a correct calling conversation.

10. Restart the **Compile and Link** job to check the FTP command and the transport of the COBOL source to your target development environment.

11. The transported COBOL source must compile in your development environment. The following picture illustrates the necessary COBOL modules and the calling hierarchy with the *R3SYSINF* function example.



Tip: Compile and link the COBOL modules with the dynamic linker option. In this case, the module *RPCSRVI* must compile only once and the same instance must be available to others.

12. Start the main module (*PR3SYSIN*) in your target environment. The main module has generated code for
    - setting the IN parameter and

- displaying the OUT parameter.

Tip: Copy the generated code of working storage section, the (in- and out-) parameter and the call statement to your business program.

Write the First Client

Technical Parameter

Each calling function has a technical parameter `RFC_SYSTEM` which contains the following fields.

```
1 RFC_SYSTEM         In Out
    2 userid      (A12)                      /* In
    2 passwd      (A8)                       /* In
    2 client      (A3)                       /* In
    2 ok          (L)                        /* Out
    2 message     (A250)                     /* Out
    2 operation   (I2)                       /* In
    2 handle      (I4)                       /* In Out
    2 transaction (A24)                      /* Out
```

The first 3 fields (`userid`, `passwd` and `client`) are used to identify the client in the SAP R/3 application server.

The field `ok` returns true or false if the call was successful. If false and an exception is thrown, the `message` field will contain the error text.

The `operation` field contains a flag indicating what the kernel process will do in the next call. The operation values can be added to perform multiple steps in a single call. If the connection to SAP is not closed, the `handle` field will return the RFC handle ID. You must use this handle ID in the next call to work with the remembered context on the R/3 application server side. For example, if you perform a query, the query context (result set) is remembered on the handle ID. This ID is necessary to navigate in the results.

The `transaction` field returns the ID for asynchronous RFC.

Overview of Operations

The operation field can contain the following values:

Operation	Value	Description
OPEN	1	Opens the communication to the R/3 application server and returns an RFC handle.
CREATE_TABLES	2	Creates the RFC tables to receive the contents.
WRITE_TABLES	4	Puts the contents of (RPC) data into the RFC table.
CALL	8	Calls the RFC function.
READ_TABLES	16	Gets the RFC table data and puts it into the RPC buffer.
FREE_TABLES	32	Frees the memory of RFC table data.
CLOSE	64	Closes the RFC handle communication.
DELETE_TABLES	128	Frees memory of RFC table data and deletes the table handle.
CALL_TRANSACTION	256	Calls the RFC function with transaction mode.

The following table shows predefined field constancy depending on programming language.

Natural defined in RFCSYS-L	COBOL defined in working-storage section
OP_OPEN	OP-OPEN
OP_CREATE_TABLES	OP-CREATE-TABLES
OP_WRITE_TABLES	OP-WRITE-TABLES
OP_CALL	OP-CALL
OP_READ_TABLES	OP-READ-TABLES
OP_FREE_TABLES	OP-FREE-TABLES
OP_CLOSE	OP-CLOSE
OP_DELETE_TABLES	OP-DELETE-TABLES
OP_CALL_TRANSACTION	OP-CALL-TRANSACTION

Simple Call

The `operation` field for a simple call to an RFC function must be set to OPEN+CALL+CLOSE.

Writing Tables

The communication client can send large amounts of data. The first call OPENs the communication and CREATEs tables. It is possible in the first call and in subsequent calls (WRITE_TABLES) to fill the table with data. When the table has been filled, the RFC function must be called with operation CALL.

Reading Tables

After calling (CALL) the RFC function, the client can receive the table data and any subsequent calls with the operation READ_TABLES. The communication can be closed and the memory can be freed up with the operation DELETE_TABLES plus CLOSE.

RFC Handle and Connection Pooling

From the OPEN call until the CLOSE, the `handle` field must contain the same value. This value must not be changed by the client program.

If a client program closes the handle, the RFC handle will not be closed by the kernel process. It will be closed later if this handle cannot be reused within a specified amount of time (timeout). The same feature is used if the client neglects to close the handle or if the client abends.

Table Count, Offset and Fill Parameter

Every table has the parameters `count`, `offset` and `fill`. Each parameter has the table name as prefix.

Parameter	Direction	Description
Count	Out	Returns the whole number of records in the RFC. table
Offset	In	Offset (index) pointer to the working record in the RFC table.
Fill	In Out	Number of filled records in the RPC multiple group. Set `-1` to use the maximum defined in IDL.

The client program that reads or writes a table must set the offset to the beginning of the record index using the `offset` parameter. In subsequent calls, the `offset` parameter must be incremented with the number of filled records.

The fill parameter is set by the client that transports multiple records from the RPC to the RFC on the operation code WRITE_TABLES. This parameter also returns the real number of transported records on the operation code READ_TABLES.

Calling Scenarios

This section explains the usage of table parameters and operation codes in abstract examples.

- All-in-one Call
- Reading Tables for Receiving Data
- Writing Tables for Sending Data
- Handle transactions
- Searching with alphanumeric keys
- Optional date field

Note: You must initialize the input parameter OFFSET to `0` if your client programming language does not automatically support this feature. The initialized value `0` is the same as `1`.

All-in-one Call

It is possible to perform all communication steps in one call. After this call it is not possible to retrieve more information, for example about the result list.

```
/* Set sending table records
MY_INPUT_TABLE_FILL         := 1            /* Send one record
MY_INPUT_TABLE[ 1 ].myField := 'Hello'      /* Set value

/* Initialize receiving table
MY_OUTPUT_TABLE_OFFSET      := 1            /* Read the first record
MY_OUTPUT_TABLE_FILL        := 10           /* Reply 10 records as result, if ↩
possible

/* Set operation code and call SAP function
RFC_SYSTEM.operation := OP_OPEN + OP_CREATE_TABLES +
                        OP_WRITE_TABLES + OP_READ_TABLES +
                        OP_CALL + OP_CLOSE
callSAP ( RFC_SYSTEM, ... )

if RFC_SYSTEM.ok then
   /* SAP has answered successfully
   write 'SAP have found number of records: ' MY_OUTPUT_TABLE_COUNT
   for 1 to MY_OUTPUT_TABLE_FILL
       /* Print all received records
   endfor
else
   /* Do error handling
endif
```

Reading Tables for Receiving Data

This scenario sends data in multiple calls.

```
/* Initialize receiving table
MY_OUTPUT_TABLE_OFFSET := 1                 /* Read the first record
MY_OUTPUT_TABLE_FILL   := 10                /* Reply 10 records as result, if ↩
possible

/* Set operation code and call SAP function
RFC_SYSTEM.operation := OP_OPEN + OP_CREATE_TABLES + OP_CALL +
                        OP_READ_TABLES
callSAP ( RFC_SYSTEM, ... )

/* Remember the context in RFC_SYSTEM.handle for all subsequent calls

if RFC_SYSTEM.ok then
   /* SAP has answered successfully
   write 'SAP has found number of records: ' MY_OUTPUT_TABLE_COUNT
   while RFC_SYSTEM.ok and MY_OUTPUT_TABLE_FILL > 0

      /* Prepare call to get next package
      MY_OUTPUT_TABLE_FILL   := 10           /* Reply 10 records as result, if ↩
possible
      RFC_SYSTEM.operation   := OP_READ_TABLES
      callSAP ( RFC_SYSTEM, ... )

      for 1 to MY_OUTPUT_TABLE_FILL
          /* Print all received records
      endfor

      /* Evaluate next OFFSET
      MY_OUTPUT_TABLE_OFFSET := MY_OUTPUT_TABLE_OFFSET + MY_OUTPUT_TABLE_FILL

   endwhile

   /* Close communication
   RFC_SYSTEM.operation := OP_CLOSE
   callSAP ( RFC_SYSTEM, ... )
else
   /* Do error handling
endif
```

Writing Tables for Sending Data

If you have a large result list then you must perform multiple calls.

```
/* Set operation code for opening communication
RFC_SYSTEM.operation := OP_OPEN + OP_CREATE_TABLES
callSAP ( RFC_SYSTEM, ... )
if not RFC_SYSTEM.ok then
   /* Do error handling and abort
endif

/* Remember the context in RFC_SYSTEM.handle for all subsequent calls

while not All_Data_Transfered_Flag
 /* Send 2 records per call
 MY_INPUT_TABLE_FILL         := 2            /* Send 2 records
 MY_INPUT_TABLE[ 1 ].myField := 'Hello'      /* Set value
 MY_INPUT_TABLE[ 2 ].myField := 'World'      /* Set value

 /* Set operation code for sending data
 RFC_SYSTEM.operation := OP_WRITE_TABLES
 callSAP ( RFC_SYSTEM, ... )
endwhile

/* Set operation code for calling SAP function and closing communication
RFC_SYSTEM.operation := OP_CALL + OP_CLOSE
callSAP ( RFC_SYSTEM, ... )
if not RFC_SYSTEM.ok  then
   /* Do error handling
endif
```

Handle transactions

If the server functions require transaction handling, the client should call `BAPI_TRANSACTION_COMMIT` or `BAPI_TRANSACTION_ROLLBACK`. Both functions must be added to the **IDL**.

```
   /* Set sending table records
MY_INPUT_TABLE_FILL         := 1            /* Send one record
MY_INPUT_TABLE[ 1 ].myField := 'Hello'      /* Set value

/* Set operation code and call SAP function
RFC_SYSTEM.operation := OP_OPEN + OP_CREATE_TABLES +
                        OP_WRITE_TABLES + OP_CALL

callSAP ( RFC_SYSTEM, ... )

/* Remember the context in RFC_SYSTEM.handle for all subsequent calls

if RFC_SYSTEM.ok then
```

```
   /* SAP has answered successfully

   /* Delete the tables in this memory context
   RFC_SYSTEM.operation := OP_DELETE_TABLES
   callSAP ( RFC_SYSTEM, ... )

   /* Commit transaction on server side
   /* Pass remembered RFC_SYSTEM.handle
   RFC_SYSTEM.operation := OP_CALL
   callSAP_COMMIT ( RFC_SYSTEM, ... )

   /* Close RFC communication
   RFC_SYSTEM.operation := OP_CLOSE
   callSAP_COMMIT ( RFC_SYSTEM, ... )
else
   /* Do error handling
endif
```

Searching with alphanumeric keys

Some keys are defined as numeric fields in SAP's database. But the BAPI interface defines this field as alphanumeric. In this case, the field must be set with leading `0` and alphanumeric.

```
MATERIAL.MATERIAL_ID = '0000000001234567890'
```

Optional date field

Some parameters are optional and have SAP's date type. Initialize a date field optionally with the value `'00000000'`

```
VALIDATE.START_DATE = '00000000'
```

Using XML RPC Client

SAP R/3 Gateway supports the generation process for creating an **XML RPC client**. Your client must send an XML document to the EntireX XML RPC adapter servlet via HTTP. The EntireX XML RPC adapter servlet can run inside the IDoc XML Gateway. A new template (since version 2.3.1.06) generates

- an XML document or
- XSL stylesheets

for the RPC protocol.

To configure the development

1 Go to the administrator developer page **Configuration**, **Development** and **Rpc2Rfc (SAP Server)**.

2 To browse the generated files, **add a link** to the developer page. Set the following values:

Description	Browse XML RPC Documents
URL	../Rpc2Rfc/dev/xml_rpc

3 Click the link **Makefile** to open the makefile in an editor. For Windows, add the following line (to the includes section):

```
include xml_rpc_win.mak
```

For UNIX, add the following line:

```
include xml_rpc_unx.mak
```

Save the changes.

4 To call the included makefile step, add the target `XML_RPC` to the `all` target line. For Windows, call the link **Makefile** in the editor:

```
all:  $(IDL_FILE_TMP) PDA $(D_ERX_LIB) $(ERX_LIB) COBOL XML_RPC
```

For UNIX, call the editor with **Include Makefile** and add the target:

```
all:  $(IDL_FILE_TMP) PDA $(D_ERX_LIB) $(ERX_LIB) COBOL XML_RPC
```

Save the changes.

5 For Windows enter the value `xml_rpc_win.mak` in the line **XML RPC Makefile**. For UNIX target operating systems enter `xml_rpc_unx.mak`. Save the changes.

How to configure the generation process

1 As default, the following 2 lines in the *xml_rpc_xxx.mak* file generate the XML and XSL files in the sub-directory *xml_rpc*.

```
$(ERXIDL) -o xml_rpc -D STYLE=XML -t xml_rpc.tpl $(IDL_DIR)/$(ERX_LIB).idl
$(ERXIDL) -o xml_rpc -D STYLE=XSL -t xml_rpc.tpl $(IDL_DIR)/$(ERX_LIB).idl
```

2 After XML and XSL files have been generated, the **EntireX Workbench** will be called in **batch mode** to generate the XMM mapping file. The resulting file has the name *R3RPC.xmm*.

3 Include the generated mapping file in the configuration of the EntireX XML RPC adapter servlet. To do this, copy the mapping file to the directory *sapr3idocxmlgateway/config* and rename it (with prefix `D`) because this file already exists.

4 Change the EntireX XML RPC adapter servlet configuration (for **Windows** or **UNIX**) *sapr3idocxmlgateway/config/AdapterConfig.xml* to include the new mapping file with the prefix `D`:

```
...
      <exx-xmm>/config/DR3RFC.xmm</exx-xmm>
...
```

5 For the changes to the EntireX XML RPC adapter servlet configuration or the mapping file to take effect, the servlet must be restarted. For example with the Tomcat web application server, use the Tomcat Manager.

8 Develop an SAP Call to an External Application (Rfc2Rpc Kernel)

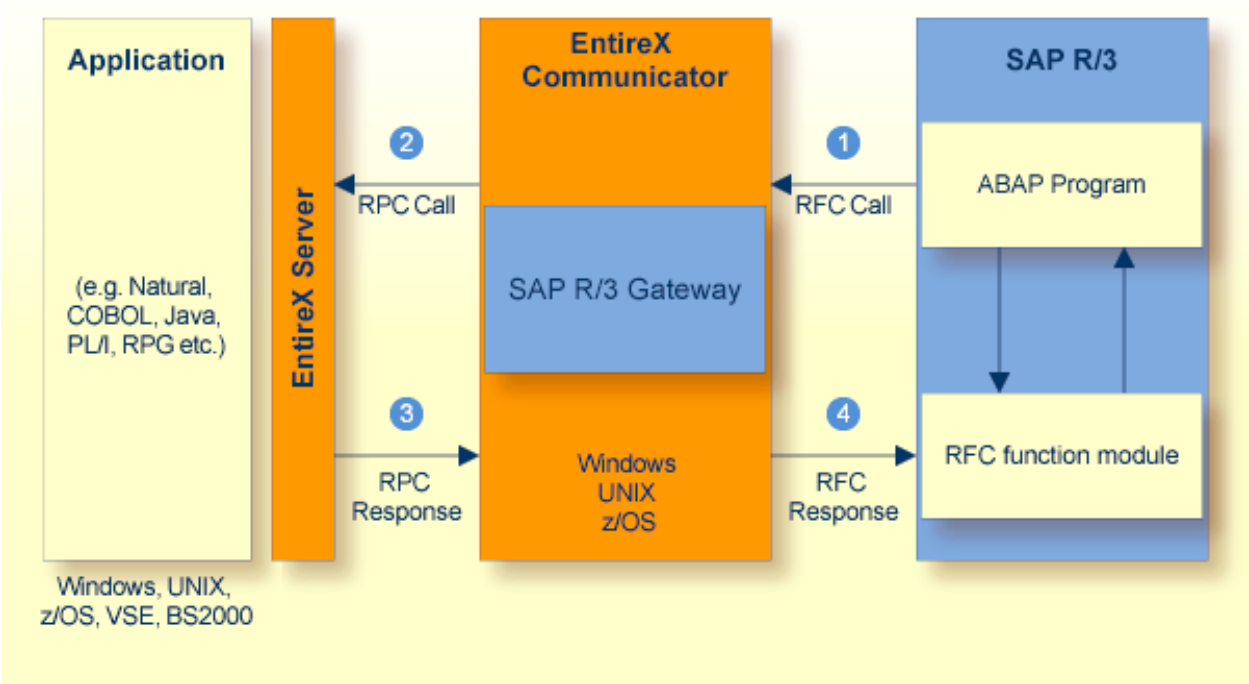
This chapter describes how you can program a connection from SAP R/3 acting as a client to an webMethods EntireX RPC server. This connection is used when R/3 initiates the communication.

The following topics are covered here:

Overview

With the webMethods EntireX Rfc2Rpc gateway, it is possible to connect SAP R/3 as client to an webMethods EntireX RPC server. In this scenario, R/3 "pushes" information to an external application.

The following figure illustrates communication flow.



Communication between the processes is provided by TCP/IP. With webMethods EntireX, this is executed using a Remote Procedure Call (RPC), and for SAP R/3 using a Remote Function Call (RFC).

The Rfc2Rpc kernel transforms the RFC call to an webMethods EntireX RPC call. Thus, subroutines in webMethods EntireX (for example, Natural) can be called by subroutine reference from SAP R/3 APAB. For R/3, the Rfc2Rpc kernel process acts as a server, and for webMethods EntireX as client.

Both RPC and RFC logs require that in both the client and the server the structure of the parameters such as data type, input or output is submitted. This section also describes the structure of the parameters in the webMethods EntireX IDL.

To develop this scenario:

- Define the interface between ABAP and webMethods EntireX in the IDL (Interface Definition Language)
- Compile and link the IDL to generate the executable kernels
- Browse to the generated ABAP client report for calling the remote service.
- Check the webMethods EntireX Broker Attribute File

These steps are described in detail in the following sections, using a calculator program as an example.

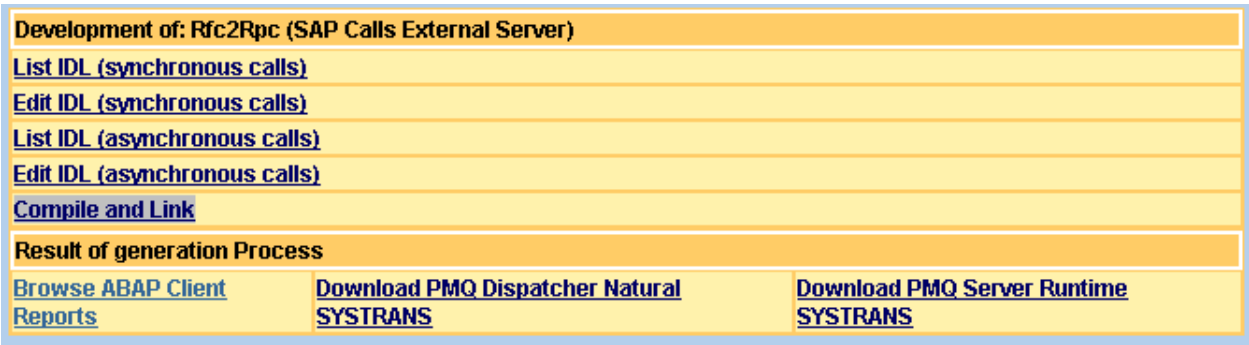
Develop IDL

For the Rfc2Rpc kernel, you must develop an IDL. In this IDL, you must define the interface between APAB and the webMethods EntireX subprograms. You can generate the contents of this IDL in any of the following ways:

- using the **IDL Extractor for Natural** within webMethods EntireX
- using the **IDL Extractor for COBOL** within webMethods EntireX from the COBOL source code.

The webMethods EntireX documentation contains more information about the **Interface Definition Language**.

All of the development steps are started from the corresponding **Development** page of the **Configuration** menu.



- Using Function Names
- Support of Import and Export Parameters
- Support of Data Types

- Support of Tables
- Workshop for Groups, Fields and Arrays
- ABAP Calls External Function
- Asynchronous Communication

Using Function Names

The example IDL file illustrated below contains the RFC function Z_EXX_CALC.

```
Library 'RFCRPC' Is

 Program 'CALC' : 'Z_EXX_CALC' Is
   Define Data Parameter
    1 Op          In
      2 Arg1   (A1)
      2 Op1    (I4)
      2 Op2    (I4)
    1 Rc         Out
      2 Furesult    (I4)
   END-DEFINE
```

The name of a subroutine in the ABAP can have up to 32 characters. To maintain the platform and language independence of the protocol layer, webMethods EntireX supports only 8 characters. The ABAP name must therefore be mapped to an webMethods EntireX name. In the above example, CALC is the subroutine name on the server and Z_EXX_CALC is the subroutine name in the ABAP. In this way, all of the programming conventions on the client and the server side can be supported.

This conversion is likewise made in the IDL in the printout program. The webMethods EntireX IDL supports a library concept. A library is introduced with the printout LIBRARY followed by a name. The use of the printout LIBRARY is described in the development environment. All subsequent printout programs within the IDL then belong to this library.

Support of Import and Export Parameters

An import or export parameter is a group without multiple occurrences. Therefore, if you have a group defined in the IDL, this must be either IN or OUT. INOUT is possible too. In this case, this group is an import and an export parameter.

A group has 1 in the IDL on the name level and several fields on level 2. The sequence of the fields on level 2 must be the same in the client as in the server. In the ABAP, a parameter is defined by the name. The names of the parameters must therefore be the same in the client as in the server on level 1.

Support of Data Types

The following **data types** subsets are supported in the IDL for the Rfc2Rpc kernel:

IDL Data Type	Description	ABAP Data Type
A*number*	String with fixed length	(*number*) TYPE C
N*number of before digits[.number of after digits]*	Unpacked decimal. The after digits information is lost in ABAP. For example with data type N2.2, setting the numeric field to 100 yields a value of 1.	(*sum of before and after digits*) TYPE N
P*number of before digits[.number of after digits]*	Packed decimal. The internal representation byte length is defined in ABAP language, while external representation is used in webMethods EntireX. To get the evaluated internal representation length, check the generated source ABAP report.	(*internal representation length of before and after digits*) TYPE P DECIMALS *number of after digits*
L	Logical. Set the value `X` in the ABAP report to transport the value `true`. Otherwise `false` is used.	(1) TYPE C
I4	Integer.	(4) TYPE I

Support of Tables

Tables in the ABAP are multiple groups. The RFC log supports the transfer of as many developments within the table as desired. However, an upper limit must be input to the IDL. If the client sends more than defined in the IDL, the corresponding webMethods EntireX function is called multiple in a conversation. For example, the table array defines 10 records. The SAP functions send 11. The webMethods EntireX function is called with 10 and with 1 record. The first call opens the conversation and the last call closes the conversation. The conversation is required to hold the contect on server side of all receiving data.

Workshop for Groups, Fields and Arrays

The following section provides a summary of the last 2 abstracts and explains a straight-forward IDL development with some rules. The rules affect the functionality of the RFC protocol.

1. Define a group with name on level 1. The name defined here must be used in ABAP's **CALL FUNCTION statements**.
2. Define an explicit direction `IN` or `OUT` for this group. `IN OUT` is possible and this parameter must be defined twice in the IMPORTING and EXPORTING statement.
3. If this group has more than one occurrence, then define the maximum as multiple group. Now this group is an RFC table.
4. Define all fields with a type only on level 2 for a specific group.

5. Define a fixed array size for fields on level 2. See generated ABAP report to pass the fixed array via the RFC protocol.

The following abstract example shows all of the possibilities for group definition:

```
...
  Define Data Parameter
    1 MY_GROUP_IN          In /* Defined 'in' group for server, exporting in ABAP ↩
caller
      2 myField  (A10)
      2 myArray     (A10/1:5) /* Fixed array
      ....
    1 MY_GROUP_OUT        Out /* Defined multiple 'out' group for server, importing ↩
in ABAP caller
      2 myField  (A10)
      2 myArray     (A10/1:5) /* Fixed array
      ...
    1 MY_MULTIPLE_GROUP_IN  (/1:10) In /* Defined multiple 'in' group for server, ↩
send table in ABAP caller
      2 myField  (A10)
      2 myArray     (A10/1:5) /* Fixed array
      ...
    1 MY_MULTIPLE_GROUP_OUT  (/1:10) Out /* Defined 'out' group for server, receive ↩
table in ABAP caller
      2 myField  (A10)
      2 myArray     (A10/1:5) /* Fixed array
      ...
  End-Define
```

ABAP Calls External Function

When developing the IDL, you defined a function in the IDL. During the **compilation and linking process**, an ABAP example report is created. This report shows the call to the defined external function. Use **Browse ABAP Client Reports** on the **Develop** page to display the generated code.

The data types for importing and exporting, and the tables are generated from the interface definition in IDL. The following example shows the report from the calculator example.

```
REPORT Z_EXX_CALC.

DATA: I TYPE I.

***************************
* Import, Export and Tables
***************************
DATA: BEGIN OF Op,
  Arg1(1)  TYPE C,
  Op1(4)  TYPE I,
  Op2(4)  TYPE I,
END OF Op.
```

```
DATA: BEGIN OF Rc,
  Furesult(4)  TYPE I,
END OF Rc.

****************************
* Set export variables
****************************
Op-Arg1 = ' '.
Op-Op1 = 0.
Op-Op2 = 0.
Rc-Furesult = 0.

****************************
* Call external Z_EXX_CALC
*
* Remove '*' in lines
* for using transactional
* and asynchronous RFC
****************************
CALL FUNCTION 'Z_EXX_CALC'
* IN BACKGROUND TASK
 DESTINATION 'EXX_GATEWAY'
 EXPORTING
  Op = Op
 IMPORTING
  Rc = Rc.

* CALL FUNCTION 'START_OF_BACKGROUNDTASK'
*  EXPORTING
*  STARTDATE = sy-datum
*  STARTTIME = sy-uzeit.
*
* COMMIT WORK.
```

The report contains all of the structures required for data transfer and the statement CALL FUNCTION with DESTINATION. The Rfc2Rpc kernel has been registered with the SAP R/3 Application Server to receive this call.

It is also possible to call the external function with a transactional RFC. This is possible only if data are sent and not received. You must activate the statements IN BACKGROUND TASK, CALL FUNCTION 'START_OF_BACKGROUNDTASK' and COMMIT WORK for a transactional RFC. The advantage of a transactional RFC is that the SAP R/3 Application Server queues these calls if no external Rfc2Rpc kernel is accessible.

Asynchronous Communication

If SAP R/3 is the leading system for master data and wants to send the data without any acknowledgement, asychronous communication is preferred. You can develop an IDL (*PMQ.idl*) for an asynchronous interface. This IDL contains the business interface between applications.

```
Program 'MESSAGE' : 'Z_EXX_MESSAGE' Is
   Define Data Parameter
    1 SEND          In
      2 field1    (A250)
END-DEFINE
```

Only IN parameters can be defined in this IDL. The Rfc2Rpc kernel generates an asynchronous message (Unit of Work) for the Persistent Message Queue (PMQ) in the webMethods EntireX broker from this information at runtime. The Unit Of Work (UOW) is needed to send the date in transactional mode. Later the message consumer also uses this feature for delivery. The message format (XML or plain text) can be defined as a parameter.

History of the IDL File

During each of the IDL file's save steps, the file is copied into a separate sub-directory with a current time stamp. This **Development** page from the **Configuration** menu (*http://YourGateway:8080/sapr3gateway/manager/devAdmin?cfg=2*) contains the backup template with the specific sub-directory information.

Compile and Link

Choose **Compile and Link** from the Develop page to start the makefile. The executable's kernels are generated from the synchronous and asynchronous **IDL**. On the results page, you will see the job output of this process. Please check that the job output report has the response code 0.

To activate the newly-generated version for the runtime environment, you must **deploy the executables** and restart the Running Task (see ***Running Task Rfc2Rpc (SAP calls External) Kernel Environment***).

By default the internal data between the RFC and RPC protocol is exchanged via stack memory; this can cause problems when large amounts of data are involved. The following parameter allocates static memory:

```
-D COMBUFFER=static
```

Add this parameter in Makefile of *http://YourGateway:8080/sapr3gateway/manager/devAdmin?cfg=2* in the command line of `ERXIDL`.

webMethods EntireX Broker Attribute File

The **webMethods EntireX Broker Attribute File** contains the resource definition for asynchronous communication. For example, there are some parameters for the Persistent Store (PSI) and the lifetime of Units of Work (UOWs).

The necessary parameters are described in a **list**.

ABAP Programming Hints

In the R/3 programming language, the call to the webMethods EntireX subroutine with the printout CALL is given in FUNCTION. The import, export and table parameters are submitted with the call. In order to route the subroutine reference to the agent (rather than remaining within the R/3 application server), `DESTINATION <target name>` must be added to the above printout. The target name is defined and assigned by the R/3 administrator. This takes place in the transaction **SM59**.

The agent will throw an RFC exception if it cannot establish a connection to the webMethods EntireX Broker. This condition should be intercepted by the calling ABAP program, since otherwise it aborts. The RFC exception is returned to the calling program as text. The exception text consists of prefix, error class and error number.

Prefix	'EXX'
Error class	webMethods EntireX Error class, 4 digits, with padded 0
Error number	webMethods EntireX Error number, 4 digits, with padded 0

Table of most frequent exceptions.

EXX02150148	webMethods EntireX Broker not reachable
EXX00070007	RPC Server not started
EXX00740074	RPC Server does not answer in the prescribed time.

Additional Parameters in IDL

It is possible to define additional parameters in the IDL file for generating UOWs. For example, the SAP client can define the queue name in webMethods EntireX Broker.

```
Program 'MESSAGE' : 'Z_EXX_MESSAGE' Is
   Define Data Parameter
    1 EXXIN         In
      2 server_name (A32)
    1 SEND          In
      2 field1    (A250)
END-DEFINE
```

If the ABAP client calls this external function, it overwrites the server name of the runtime parameter EXX_PMQ_SERVER_ADDRESS. Note that the EXXIN parameter is not sent to the RPC Server. Using the following fields, you can control how messages are sent to the webMethods EntireX Broker.

Fieldname	Description
server_name	Overwrites the server queue name of EXX_PMQ_SERVER_ADDRESS.
broker	Overwrites Broker ID address of EXX_PMQ_SERVER_ADDRESS.
lifetime	Defines unit of work lifetime.
do_xml	Sends message in XML format.
do_rpc	Sends message in RPC format.
do_not_uow_query_last	Do not get previous conversation ID. Always create a new conversation ID.
waittime	Wait this length of time to receive an answer.
do_new_conversation	Enables the client to control the conversation ID. Set `do_new_conversation = 'T'` to create always a new conversation for each UOW. Other value does a query for old existing conversation.

9 Overview of Running Tasks

All kernel tasks are maintained on the **Running Tasks** page of the **Worker** menu (*http://YourGateway:8080/sapr3gateway/manager/run*). For information about the SAP R/3 Gateway GUI and how to use System Manager, see the section ***System Manager***.

The following figure illustrates the **Running Tasks** page. The first column shows the instance of a Running Task. The link points to parameters of an

- **Rpc2Rfc kernel** or
- **Rfc2Rpc kernel**

Running Task	Operation	Startup	Status	Start at	Stop at	Output	Resource
Rpc2RfcServerDev	Start	attach	Task has stopped	2004-02-25 12:13:23	2004-02-25 15:33:42	Log	Shutdown Restart
Rpc2RfcServerInt	Stop	attach	Task has started	2004-01-22 08:27:42		Log	Shutdown Restart
Rpc2RfcServerProd	Stop	attach	Task has started	2004-01-22 08:27:45		Log	Shutdown Restart
Rfc2RpcServerDev	Stop	attach	Task has started	2004-02-26 11:43:13		Log	
Rfc2RpcServerInt	Start	attach					
Rfc2RpcServerProd	Start	attach					
DCOMServer	Start	attach					Shutdown Restart
PMQServerDev	Start	attach					Shutdown Restart
MediatorHostManager	Start	attach					Shutdown Restart
Rpc2RfcServerDevIBM	Start	attach	Task has stopped	2004-02-24 13:38:10	2004-02-25 19:27:29	Log	Shutdown Restart
Task Controller	Set disabled		enabled				
Log filename	Set new Log		.2004-02-26				
Clean up Log Files	Show						

The tasks can be started and stopped with the button in the **Operation** column. The stop command cancels the task at the operating-system level. The start command starts a process as a child of this Application Server. The button is toggled between start/stop commands.

There are 3 startup types (the **Startup** column):

attach	There is no controller for starting and stopping.
manual	This task must be started by the user. If this task terminates abnormally, it will restart automatically.
auto	This task is started automatically when this Application Server is booted. If this task terminates abnormally, it will restart automatically.

To display more information about the parameters of a Running Task, you can choose the link in the first column.

The standard and error output is written to a file. You can show this output by choosing the link **Log** in the **Output** column. The name of this file consists of:

1. Current directory path
2. Kernel task name

3. A timestamp
4. The extension ".log"

A **Shutdown** and a **Restart** command are available for some tasks. These buttons are only available if the configuration of a task defines the shutdown command.

Shutdown	The defined shutdown command is executed and the Task Controller is disabled.
Restart	The defined shutdown command is executed and the Task Controller is unchanged. Therefore, if the startup type is set to "manual" or "auto", the task will be started immediately. This feature can be used when changing parameters. Please check the associated **Start-at** time.

In general, it is possible to disable the Task Controller for all running tasks. This is necessary if you want to stop all tasks for a period using an Operating System command or other tool. In this case, the autostart is inactive. Please do not forget to enable the Task Controller.

For more detailed information on the configuration of runnings tasks, refer to the section **Running Tasks** under the heading "Workers".

10 Running Task Rpc2Rfc (Client calls SAP) Kernel Environment

This chapter describes all of the parameters for this task. See *Overview of Running Tasks* for general information on starting and stopping running tasks.

Parameter

The following figure shows the first parameters.

Running Task Parameter for: Rpc2RfcServerDev		
Name:	Rpc2RfcServerDev	
Description:	Rpc to Rfc (SAP Client)	
Directory:	$SM_HOME/Rpc2Rfc/run/dev	
Command:	RPC_SRV cfg=Server.cfg -s	
Startup:	attach	
Sleep before automatic Restart:	0	in seconds
Filter lets through to System Log:	none	
Log Life Time:	1	in days
Config file:	$SM_HOME/Rpc2Rfc/run/dev/Server.cfg	
Shutdown Command:	exxshutdown localhost@RPC/R3RFCD/CALLNAT	
Shutdown Command Directory:	$SM_HOME/Tools	
Shutdown Command URL:	operation=exxcisShutdown&broker=localhost&service	
Shutdown Server Address:		
Shutdown User ID:		
Shutdown Password:		
RFC Destination:	EXX_RFC_DEST=sapids	
RFC Load Balancing Group:	EXX_RFC_GROUP=	optional
RFC Load Balancing R/3 Name:	EXX_SERVER_ADDRESS=local	optional
RFC Client:	EXX_RFC_CLIENT=800	
RFC User ID:	EXX_RFC_USER=cpic	
RFC Password:	**********	
SAP System Number:	EXX_RFC_SYSNR=00	
SAP Language:	EXX_RFC_LANG=DE	
RFC Trace:	EXX_RFC_TRACE=0	
Connection Pool clean up:	EXX_RFC_CLEANUP=10	in seconds
Connection Pool Time Limit:	EXX_RFC_TIMELIMIT=60	in seconds
Gateway Trace:	EXX_GW_TRACE=137	0=off 1=stdout

Description of parameters:

Parameter	Environment Variable	Description
Name		Name of the running task. The name should not have spaces and is used to evaluate log-file name.
Description		Short description of the running task.
Directory		Current directory path of the running task.
Command		Command to start the RPC Server under **Windows** or **UNIX** with configuration file a parameter.
Startup		Startup type. See *Overview of Running Tasks*.
Sleep before automatic restart		On abnormal termination, time in seconds before this task is restarted.
Filter lets through to System Log		The possible values of the filter are: "none", "all", "stdout", and "stderr". Example of "a The full output of the task is written to **System Log**.
Log Lifetime		Lifetime of log files.
Configuration File		Path and filename of the RPC Configuration file. The link starts an editor at the end o the next page. See the webMethods EntireX documentation for a detailed descriptior under **Windows** or **UNIX**. You should set the connection parameters for webMethoc EntireX Broker and the number of worker threads. The generated libraries are availa for multi-threading. This parameter can control the maximum number of concurrent parallel requests received from the clients.
External Configuration Properties		This text editor is available if the configuration file is imported with the command **Imp external configuration file**. The external RPC Server is now maintained inside **Syste Manager** and saved in **history**, if this page is saved with all parameters.
Shutdown Command		Execute this command for normal shutdown.
Shutdown Command Directory		Directory path of shutdown command.
Shutdown Command URL		HTTP get request command for shutdown. The example calls the built-in shutdown command of the System Manager: `operation=exxcisShutdown&broker=localhost&service=RPC/R3RFCD/CALL`
Shutdown Server Address	EXX_SERVER_ADDRESS	Set the server address for the RPC server to deregister on the received signal. Use the following syntax: `EXX_SERVER_ADDRESS=<brokerID>@<class>/<server>/<servi`
Shutdown User ID	EXX_USERID	Set the webMethods EntireX Broker user for deregistration.
Shutdown Password	EXX_PASSWD	Set the webMethods EntireX Broker security password. This field is saved. If your password were `hello` you would type in `EXX_PASSWD=hello`.
RFC Destination	EXX_RFC_DEST	DNS name of SAP R/3 application server.

Parameter	Environment Variable	Description
RFC Load Balancing Group	EXX_RFC_GROUP	Load balancing group name.
RFC Load Balancing R/3 Name	EXX_RFC_R3NAME	Load balancing R/3 name.
RFC Client	EXX_RFC_CLIENT	3-character client name.
RFC user ID	EXX_RFC_USER	CPIC user ID.
RFC Password	EXX_RFC_PASSWD	Password for user ID. This field is saved. If your password were `hello`, you in `EXX_RFC_PASSWD=hello`.
SAP System Number	EXX_RFC_SYSNR	2-character system number.
SAP Language	EXX_RFC_LANG	Client language.
RFC Trace	EXX_RFC_TRACE	Switch RFC trace off (=0) or on (=1).
Connection Pool Cleanup	EXX_RFC_CLEANUP	Starts connection pool cleanup thread, if value higher than 0. Asks for unused at these intervals in seconds.
Connection Pool Time Limit	EXX_RFC_TIMELIMIT	If this time limit is exceeded, the connection will be closed. The value is in se
Gateway Trace	EXX_GW_TRACE	Rpc2Rfc kernel trace.
Gateway Logon Strategy	EXX_GW_LOGON_PRIO	Handle user ID, password and client at logon time. It is possible to set the logo when the kernel is started up, or during runtime as a function call paramete (**client/userid/password**). This parameter sets the priority to `Running Task overwrites function call` or `Function call overwrites Running parameter`.
Path	PATH	Set **operating system environment variable PATH**.
webMethods EntireX TCP/IP Timeout	ETB_TIMEOUT	See **webMethods EntireX ETB_TIMEOUT** parameter definition.
webMethods EntireX Stublog	ETB_STUBLOG	See **webMethods EntireX ETB_STUBLOG** parameter definition.
webMethods EntireX Stublog Directory	ETB_STUBLOGPATH	Directory path of stublog output.
webMethods EntireX Transport Protocol Rule	ETB_TRANSPORT	See **webMethods EntireX ETB_TRANSPORT** parameter definition.

Parameter	Environment Variable	Description
Library Path	LD_LIBRARY_PATH or others	**UNIX library path**, depending on operating system.
SAG Home Directory	SAG	UNIX **SAG home** directory path.
webMethods EntireX Directory	EXXDIR	webMethods EntireX directory path.
webMethods EntireX Version	EXXVERS	webMethods EntireX version.
Environment Variable	ETBLNK	Filename of **Broker Stub shared library**.
Time Zone	TZ	Time zone for kernel.
SAP Codepage	SAP_CODEPAGE	Defines the codepage for this kernel to send and receive characters via RFC. If the communication partner defines another codepage, the RFC protocol transforms the characters. If a character cannot be translated, the current character is replaced by # (default character in this case). Use this parameter and set the codepage of your communication partner.

An overview of all webMethods EntireX environment variables is provided in the section **Environment Variables in webMethods EntireX Communicator** in the webMethods EntireX documentation.

SAP Load Balancing

If SAP Load Balancing is enabled, the administrator must set the following parameters:

EXX_RFC_R3NAME	For the name of the R/3 system.
EXX_RFC_GROUP	For the name of a group of application servers.
EXX_RFC_DEST	For the host name of the message server (MSHOST parameter on `RfcOpenEx()` function).

SAP Router

The connectivity to the SAP application via SAP router is supported and the administrator must set the following parameters:

EXX_RFC_DEST	/H/my_SAP_router/H/my_SAP_appl_server	Replace `my_SAP_router` and `my_SAP_app_server` by the IP address or DNS name of these hosts.

NAT Gateway

The connectivity to the SAP application via NAT gateway is supported and the administrator must set the following parameters:

EXX_RFC_DEST	The value is used for `ASHOST` parameter on `RfcOpenEx()` function.
EXX_RFC_GWHOST	The value is used for `GWHOST` parameter on `RfcOpenEx()` function.

EntireX Broker Parameter

The Rpc2Rfc kernel performs a register call on the webMethods EntireX Broker. The **Broker Attribute file** must have an entry in the RPC protocol for this server. For synchronous RPC communication, add the following lines to the `Defaults = Service` section

```
CONV-NONACT = 4M
SERVER-NONACT = 5M
TRANSLATION = SAGTCHA
   CLASS = RPC ,   SERVER = * ,   SERVICE = CALLNAT
```

The `SERVER = *` defines a placeholder for all Rpc2Rfc kernels.

Gateway Logon Strategy

Client connectivity needs a client, user ID and password to identify itself to the SAP application server. These 3 parameters can be set in one of two places:

- **the Rpc2Rfc kernel** or
- **the EntireX RPC client**.

You can control the priority of parameters:

- Running Task parameter overwrites function call
- Function call overwrites Running Task parameter

The next sections explain how to use this parameter.

Running Task parameter overwrites function call

The client, user ID and password parameters of the Rpc2Rfc kernel are the master and are always used. The EntireX RPC client parameters will have no effect, if these parameters have been set in the kernel. The administrator can control the parameters from a central location and the client programmers do not need to maintain the connection parameters.

The most common situation is to use these parameter settings for retrieval function calls, for example `BAPI_MATERIAL_GETLIST`. There are fewer security requirements for identifying the client and an open connection can be reused often in the cache.

Function call overwrites Running Task parameter

The Gateway logon strategy parameter should set to this value, if the client must be identifiable, with update operations for example. If the client sends an order item, the auditing department must later know which user has sent the requirements.

Leave the fields client, user ID and password emtpy (or blank), to use the default Rpc2Rfc kernel settings. You can use this logon strategy to start with retrieval functions. The update functions will pass the logon parameters from EntireX RPC client to the SAP application server later.

11 Running Task Rfc2Rpc (SAP calls External) Kernel Environment

This chapter describes all of the parameters for this task. See the ***Overview of Running Tasks*** for general information on starting and stopping running tasks.

The first start of this kernel requires some configuration in SAP R/3. See **Running Task Rfc2Rpc (SAP calls External) Configurationon**. The following figure shows the layout of the default parameter page:

Running Task Parameter for: Rfc2RpcServerDev	
Name:	Rfc2RpcServerDev
Description:	Rfc to Rpc (SAP Listener)
Directory:	$SM_HOME\Rfc2Rpc\run\dev
Command:	rfcrpc -gsapids -aEXX_GATEWAY -x3300 -g<DNS name> -a<Program ID> -x<Port number>
Startup:	attach
Sleep before automatic Restart:	0 in seconds
Filter lets through to System Log:	none
Log Life Time:	in days
EntireX Server Address:	EXX_SERVER_ADDRESS=localhost@RPC/SRV1/C
EntireX Client User ID:	EXX_USERID=D-SAP-USER
EntireX Client Password:	**********
EntireX Receive Timeout:	EXX_RCV_TIME_OUT=50 in seconds
Worker Threads:	EXX_RFC_WORKER=1
RFC Trace:	EXX_RFC_TRACE=0
EntireX PMQ Server Address:	EXX_PMQ_SERVER_ADDRESS=localhost@QUEUE
EntireX PMQ Token:	EXX_PMQ_TOKEN=D-SAP-USEI
Path:	PATH=RFC_PATH;EXX_PATH
EntireX TCP/IP Timeout:	ETB_TIMEOUT=50 in seconds
EntireX RPC Runtime Trace:	No environment= variable defined No tracing. NONE= No tracing. Standard= Incoming and outgoing user data Advanced= Standard plus EntireX RPC internal functions
EntireX Stublog:	ETB_STUBLOG=0
EntireX Stublog Directory:	ETB_STUBLOGPATH=$SM_HOI since EntireX 7
EntireX Transport Protocol Rule:	ETB_TRANSPORT=TCP

Description of parameters:

Parameter	Environment Variable	Description
Name		Name of the running task. The name should not have spaces and is used to evaluate a log-file name.
Description		Short description of the running task.
Directory		Current directory path of the running task.
Command		Command to start this kernel with the parameters ■ -g<DNS Name of SAP R/3> ■ -a<Program ID> ■ -x<Port of SAP R/3>
Startup		Startup type. See *Overview of Running Tasks*.
Sleep before automatic Restart		On abnormal termination, length of time in seconds before this task is restarted.
Filter lets through to System Log		The possible values of filter are "none", "all", "stdout", "stderr". Example "all": The full output of task is written to **System Log**.
Log Lifetime		Lifetime of log files.
webMethods EntireX Server Address	EXX_SERVER_ADDRESS	webMethods EntireX Broker Address requests are sent to: `<BrokerID:Port>@<CLASS>/<SERVER>/<SERVICE>`. **Note:** The `BrokerID` of this parameter is checked during the startup of the kernel. A logon call to the Broker is performed with the given `BrokerID`. If the Broker is not reachable, the process will be stopped. Startup types `auto` or `manual` will perform a restart later.
webMethods EntireX Client User ID	EXX_USERID	User ID for webMethods EntireX Broker requests.
webMethods EntireX Client Password	EXX_PASSWD	Set the webMethods EntireX Broker Security password. If your password were `hello`, you would type in `EXX_PASSWD=hello`.
Natural RPC Client User ID	EXX_RPC_USERID	User ID for Natural Security RPC to logon to the Natural library.
Natural Security RPC Client Password	EXX_RPC_PASSWD	Set the Natural Security password. If your password were `hello`, you would type in `EXX_RPC_PASSWD=hello`.
webMethods EntireX Receive timeout	EXX_RCV_TIME_OUT	Wait this length of time for an answer.
Worker threads	EXX_RFC_WORKER	Number of worker threads of the multi-threaded kernel to receive multiple requests from SAP R/3.

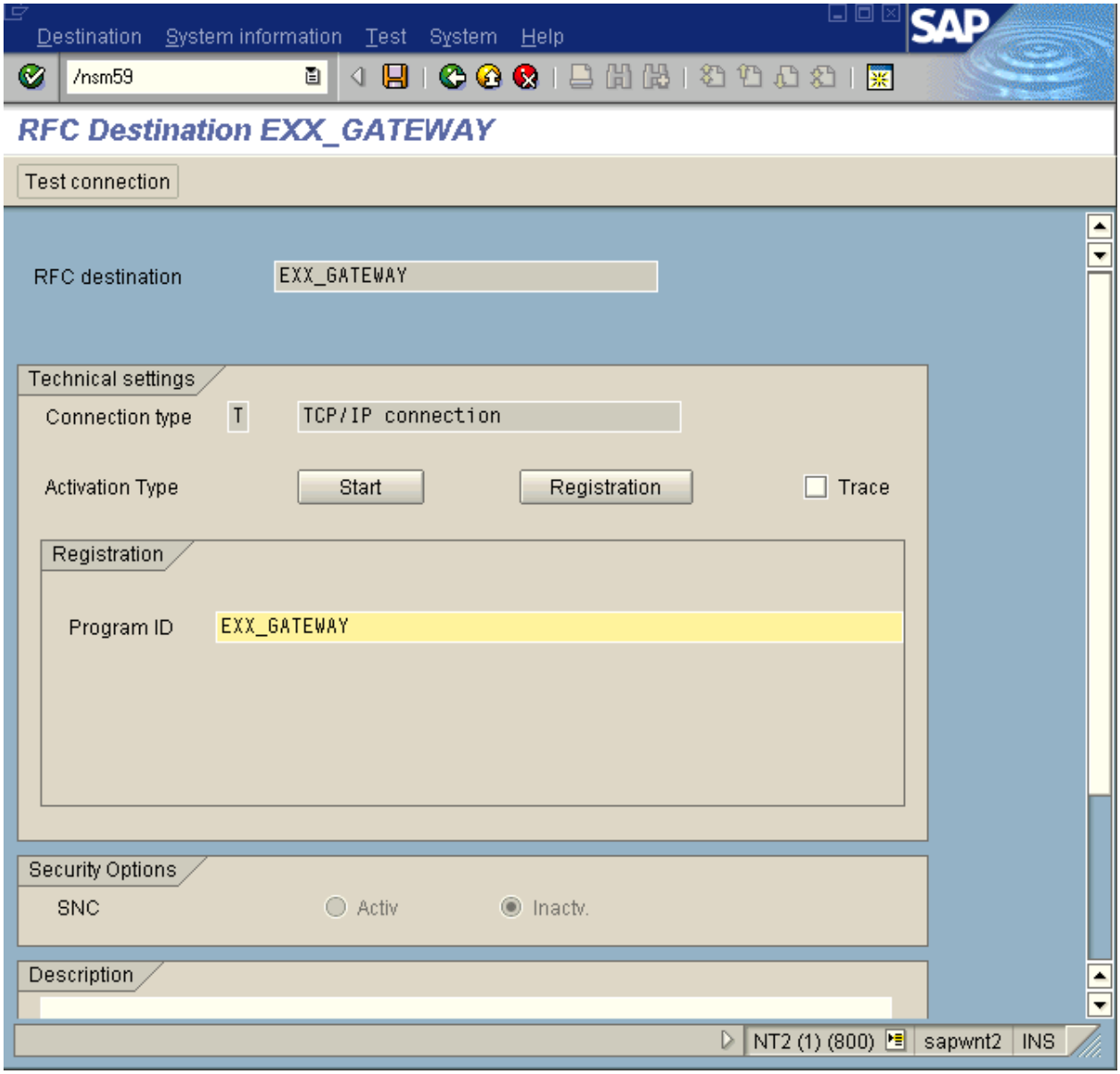
Parameter	Environment Variable	Description
RFC Trace	EXX_RFC_TRACE	Switch RFC trace off (=0) or on (=1).
webMethods EntireX PMQ Server address	EXX_PMQ_SERVER_ADDRESS	Send asynchronous units of work to this webMethods EntireX Broker address.
webMethods EntireX PMQ Token	EXX_PMQ_TOKEN	Use this token for logon connection and to query for old conversations.
Path	PATH	Set **operating system PATH** environment variable.
webMethods EntireX TCP/IP Timeout	ETB_TIMEOUT	See **webMethods EntireX ETB_TIMEOUT** parameter definition.
webMethods EntireX RPC Runtime Trace	ERX_TRACELEVEL	See **webMethods EntireX ERX_TRACELEVEL** parameter definition.
webMethods EntireX Stublog	ETB_STUBLOG	See **webMethods EntireX ETB_STUBLOG** parameter definition.
webMethods EntireX Stublog Directory	ETB_STUBLOGPATH	Directory path of stublog output.
webMethods EntireX Transport Protocol Rule	ETB_TRANSPORT	See **webMethods EntireX ETB_TRANSPORT** parameter definition.
Library Path	LD_LIBRARY_PATH or others	**UNIX library path**.
SAG Home Directory	SAG	UNIX **SAG home** directory path .
webMethods EntireX Directory	EXXDIR	webMethods EntireX directory path.
webMethods EntireX Version	EXXVERS	webMethods EntireX version.
Environment Variable ETBLNK	ETBLNK	Filename of **Broker Stub shared library**.
Time Zone	TZ	Time Zone for kernel.
SAP Codepage	SAP_CODEPAGE	Defines the codepage for this kernel to send and receive characters via RFC. If the communication partner defines another codepage, the RFC protocol transforms the characters. If a character cannot be translated, the current character is replaced by # (the default character in this case). Use this parameter and set the codepage of your communication partner.

Parameter	Environment Variable	Description
EntireX RPC Retry Calls on Error	EXX_RPC_RETRY_COUNT	Sets the number of retry calls on failed RPC request. The default is 2 if this parameter isn't set. The value 2 allows a second RPC request if the first was failed. The value 1 doesn't retry and use this setting if the called function does an update in database.

An overview of all **webMethods EntireX environment variables is on this page**.

12 Running Task Rfc2Rpc (SAP calls External) Configuration

To call subprograms from an SAP R/3 application server using the ABAP `CALL FUNCTION` statement, you must define an RFC-Destination. Use the `SM59` transaction in SAPGUI to access the following dialog. Create a new RFC-Destination of type `T` with a `Program ID`. You will reach the `Program ID` input line on pressing the **Registration** button.

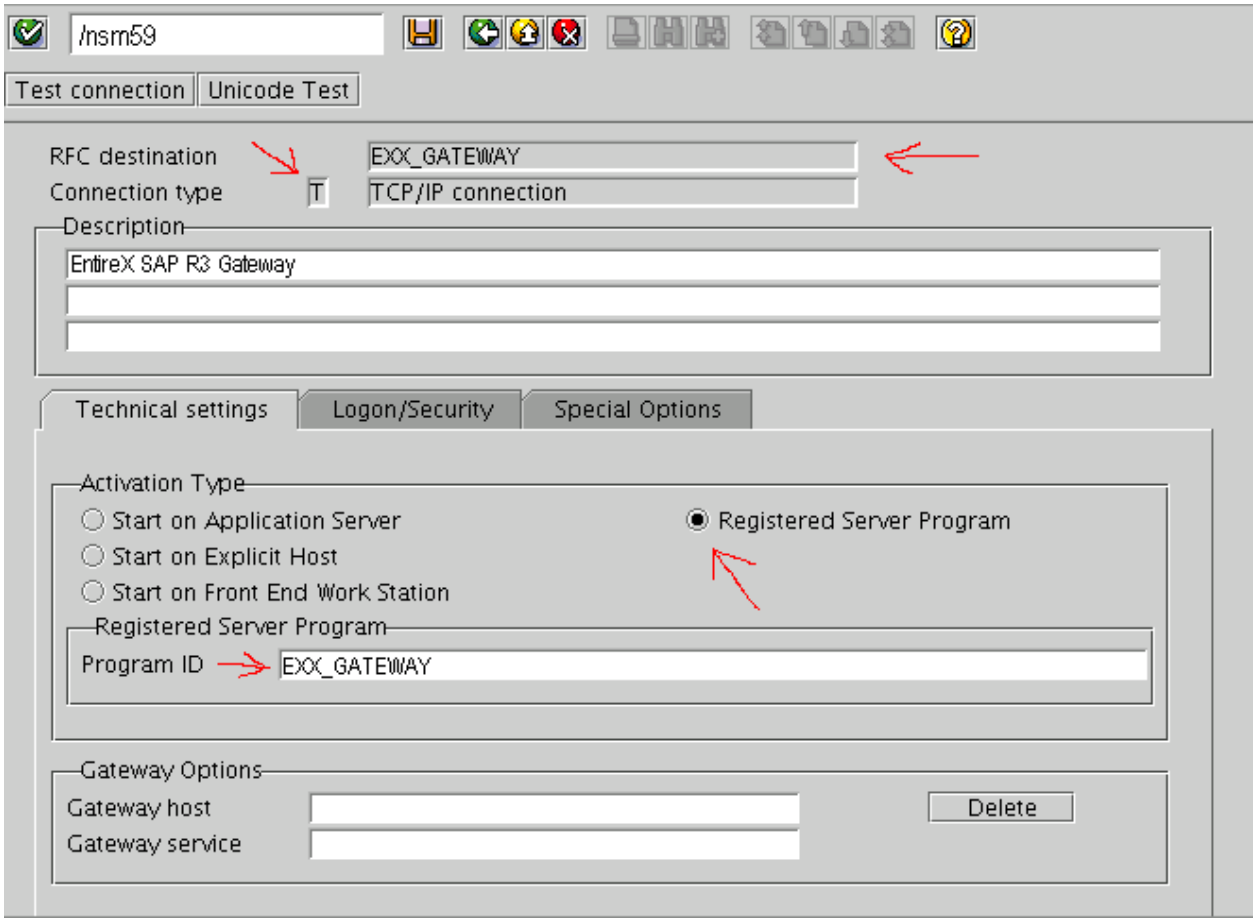


Use the name of RFC-Destination `EXX_GATEWAY` in the ABAP `CALL FUNCTION '...' DESTINATION='EXX_GATEWAY'` statement. The program ID must be set as a command line parameter (-a) for the executable Rfc2Rpc kernel. (In the example here, the program ID and RFC-destination are the same strings. You can use different names.)

Tip: After creating an RFC-Destination and starting the Rfc2Rpc kernel, you can ping the external process with **Test connection** from the R/3 application server.

Note: The program ID is case sensitive. If you change the program ID or RFC destination name, then restart the **Rfc2Rpc kernel** to perform a new registration.

The following picture shows the same transaction in SAP 4.7 and in another SAPGUI client.



Select only the **Gateway Options**, if more than one SAP application server calls the same **Rfc2Rpc kernel**. For example, in the SAP production environment there are many application servers and the RFC client program can run anywhere. Therefore, select one gateway and set the address here. Use the same address with `-g` option in **Rfc2Rpc kernel** for the gateway.

13 Deploy Kernels

The normal way of working within a programming project is to maintain different environments, to allow for development and testing without impacting production. Development with SAP R/3 Gateway allows for development, integration (or quality-testing) and production environments. First, the generated applications are tested in the development environment. After passing the tests, the kernels must be copied to the integration environment and so on. This process of copying from one environment to another is called deployment.

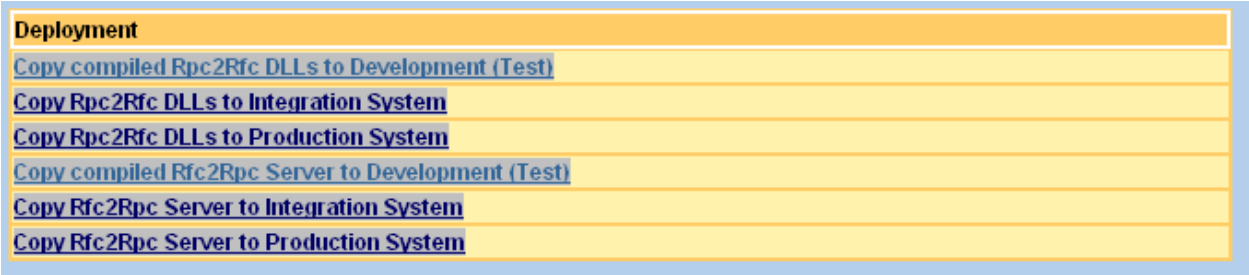
Default Deployment in the File System

The SAP R/3 Gateway supports the following default deployment process. Each environment has its own directory in the file system, depending on kernel type.

Directory	Environment	Script
<kernel type>/dev	Compile and Link	
<kernel type>/run/dev	Development	*deployFrom* copy from compile and link.
<kernel type>/run/int	Integration (Quality)	*deployFrom* copy from development.
<kernel type>/run/dev	Production	*deployFrom* copy from integration.

The *deployFrom* scripts copy the executables and IDL files from one environment to the next using operating-system commands. The IDL files are copied for documentation reasons. To search for differences in the environments, the IDL files can be compared.

All of the deployment steps can be started from the **Deployment** page of the **Tools** menu or by using the URI *http://YourGateway:8080/sapr3gateway/manager/dep*.



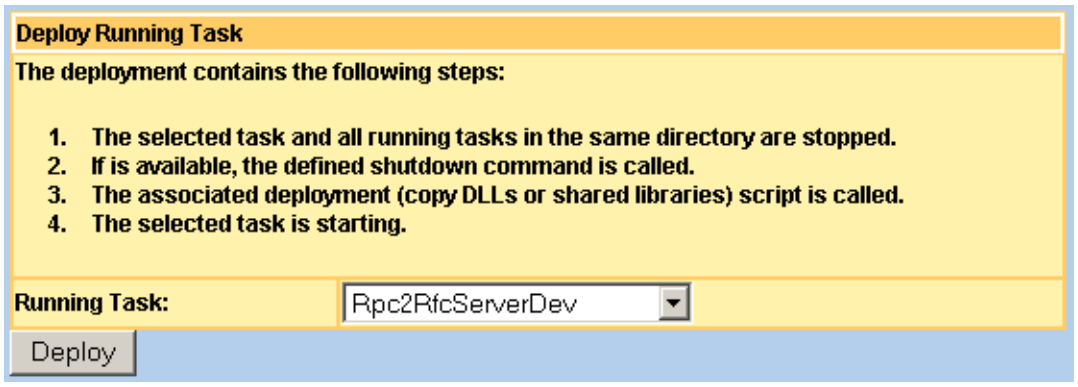
Choose the link to start the deployment process. A job is started with a batch file (on Windows) or a shell script (on UNIX) and the resulting output is displayed on an HTML page. You will see the copied files or an error message. The error message may indicate that it is not possible to overwrite executables if they are already running.

In this case, stop the Running Task (on the **Runnings Tasks** page of the **Worker** menu) or use the URI *http://YourGateway:8080/sapr3gateway/manager/run* and repeat the deployment step.

It is, of course, possible to change the default *deployFrom* scripts if you are working in a distributed environment of SAP R/3 Gateway instances. In this case, the file copy statements must be changed, for example to an FTP command. The list of all the scripts is on the **Parameter** page of the **Configuration** menu or can be viewed using the URI *http://YourGateway:8080/sapr3gateway/manager/parameter*. The link **Startup script** starts an editor on the same page.

Smart Deployment

There is a new deployment page (since 2.3.1.04) from which a number of steps are performed automatically. This page is callable by *http://YourGateway:8080/sapr3gateway/manager/deployRunningTask*. After the first call by URI, this page is appended to the end of the **Tools** menu with the item name **Smart Deployment**. You must first select the running task.



Deployment contains the following steps, which will be performed in a single submit.

1 The selected task and all running tasks in the same directory are stopped. All running tasks which use the directory are evaluated by the current configuration.

2 If the shutdown command is available for the selected task, it is called to release the resources from EntireX Broker.

3 The associated deployment script (copy DLLs or shared libraries) is called.

4 The selected task is started.

Package Builder

The package builder supports the deployment process between different application server instances. As protocol, an HTTP connection is used. The package builder creates a ZIP file (the package) and transports it (with HTTP) to the target application server. The **Upload feature** is called at the receiver (target). For example, the Rpc2Rfc kernel production environment runs on another application server. The executable files must copied between two machines.

Package Process Deployments	Operation
Copy DLLs from Dev to Int	Deploy Copy to Delete Down
Copy Rpc2Rfc DLLs from Int to Prod	Deploy Copy to Delete Up
Create new Package	

Edit Package: Copy Rpc2Rfc DLLs from Int to Prod	
Name:	Copy Rpc2Rfc DLLs from Int to Prod
Source URL:	http://integrationNode:8080/sapr3gateway/manager/xi Check Connection Example: http://integrationNode:8080/sapr3gateway/manager/xml
Source User ID:	
Source Password:	
Source Directory:	Rpc2Rfc/run/int
Selection:	*.dll; *.idl
Target Directory:	Rpc2Rfc/run/prod
Target URL:	http://productionNode:8080/sapr3gateway/setup/xml Check Connection Example: http://productionNode:8080/sapr3gateway/setup/xml
Target User ID:	
Target Password:	
Backup:	false
Smart Deployment enabled:	false
Save	

This page can be called with *http://YourGateway:8080/sapr3gateway/manager/packageBuilder*. After the first call, the **Package and Deployment** menu item is created in the **Tools** menu. After the second request, you will see the menu item.

The initiator of a deployment process is the target system. If, for example, you want to copy the executable files from the integration to the production environment, the production system is the target and the initiator. The integration system is the source.

To define a package process deployment, the following parameters are needed:

Parameter	Description
Name	Name of the deployment process.
Source URL	Defines the URL of the source system. This source system builds the package for the target. Call the manager of SAP R/3 Gateway in the source system. The URL looks like this: `http://integrationNode:8080/sapr3gateway/manager/xml`. Replace `integrationNode` with the IP or DNS name. **Tip:** Use **Check Connection** to check the HTTP connection with URL, user ID and password. The next page shows the result of the HTTP request. The source system must be implemented by the `PackageBuilder` resource. **Check Connection** checks this, too. If there is an error message, see **how to install the Package Builder resource**
Source User ID	Connect with the user ID to the source system. The HTTP Basic Authentication method is used.
Source User Password	Identify the user with this password in the source system.
Source Directory	Define the subdirectory where the package is built. The link **Source Directory** shows the contents of the file system on the next page. This directory is a subdirectory of **$SM_HOME**.
Selection	Select the files in the source directory. The wildcard character* is supported. Separate multiple selections with the `;` character. For example: `*.idl; *.dll`.
Target Directory	Define the subdirectory in the target system. The link **Target Directory** shows the contents of the file system on the next page. This directory is a subdirectory of **$SM_HOME**.
Target URL	Define the receiver of the package. The sender (this is the source system) of the package must call the **Upload and Setup component**. The URL looks like this: `http://productionNode:8080/sapr3gateway/setup/xml`. **Note:** The source system is the sender of the package. Therefore, the sender must have the IP or DNS name of the target system. (`localhost` does not work in this case.) **Tip:** Use **Check Connection** to check the HTTP connection with URL, user ID and password. The next page shows the result of the HTTP request.
Target User ID	Connect with the user ID to the target system. The HTTP Basic Authentication method is used.
Target Password	Identify the user with this password in the target system.
Backup	The value `true` creates a backup ZIP file with the old files before they are overwritten by the new ones.
Smart Deployment enabled	The value `true` **enables** this package definition for **Smart Deployment**.

Tip: Save the settings before a link for checking or directory listing is used.

The following operations are available for the list:

Operation	Description
Deploy	Start package processing and deployment. The result of the process is displayed on the next page.
Copy to	Copy this package process definition and add the new one to the end of the list.
Delete	Delete this package process definition from the list.
Up	Set this before the previous one.
Down	Set this after the next one.

If you start the package and deployment process with the **Deploy** button, the next page shows the result of updated or created files.

Result of Deployment: Copy Rpc2Rfc DLLs from Int to Prod			
Rpc2Rfc/run/prod/DR3RFC.dll	updating	2005-06-01 00:22:32	81920
Rpc2Rfc/run/prod/DSYSTEM.dll	updating	2005-06-01 00:22:32	81920
Rpc2Rfc/run/prod/R3RFC.dll	updating	2005-06-01 00:22:34	180224
Rpc2Rfc/run/prod/R3RFC.idl	updating	2005-02-12 17:40:10	15968
Rpc2Rfc/run/prod/SYSTEM.dll	updating	2005-06-01 00:22:34	180224

How to enable Package Builder for Smart Deployment

Smart deployment can start the **package builder**. In this case, the **local default deployment** is switched off. In smart deployment you can select the replaced running task, the associated package is built in the source environment and deployed in the target environment. The following output shows the result of package builder during smart deployment. The running task `Rpc2RfcServerProd` is selected for deployment and the package `Copy Rpc2Rfc DLLs from Int to Prod` is processed.

Running Task	Status	Start at	Stop at
Rpc2RfcServerProd	Task has stopped	2005-09-01 14:40:17	

Job Output - Return Value: 0

Job Output - Return Value: 0

Result of Deployment: Copy Rpc2Rfc DLLs from Int to Prod			
Rpc2Rfc/run/prod/DR3RFC.dll	updating	2005-06-01 00:22:32	81920
Rpc2Rfc/run/prod/DSYSTEM.dll	updating	2005-06-01 00:22:32	81920
Rpc2Rfc/run/prod/R3RFC.dll	updating	2005-06-01 00:22:34	180224
Rpc2Rfc/run/prod/R3RFC.idl	updating	2005-02-12 17:40:10	15968
Rpc2Rfc/run/prod/SYSTEM.dll	updating	2005-06-01 00:22:34	180224

Running Task	Status	Start at	Stop at
Rpc2RfcServerProd	Task is starting		

How the system works

1 **Smart Deployment** searches the package in the configuration environment.

2 The directory of the selected running task (e.g. **Rpc2Rfc kernel**) must be the same as the **Target Directory** of the package.

3 The flag for **Smart Deployment enabled** must also be enabled.

4 Now **Smart Deployment** calls the **Package Builder** instead of the **local default deployment**.

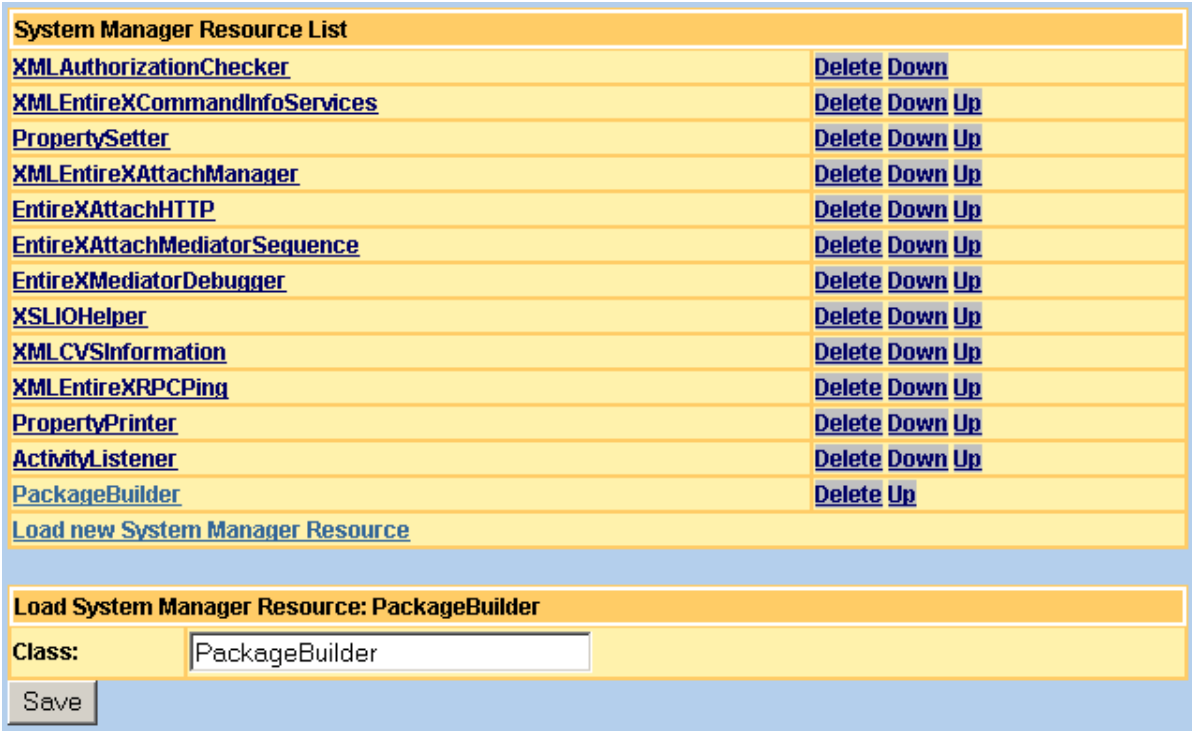
How to install the Package Builder

Every source system that builds a package for deployment must configure the Package Builder resource. As default, the package builder is not installed. Perform the following steps only for a source system. The target system, the receiver of packages, does not need the package builder.

Add Package Builder as Resource

1 Call the resource administrator page *http://YourGateway:8080/sapr3gateway/manager/resourceAdmin* on the source system.

2 Create and add a new resource, if the class `PackageBuilder` is not available.



System Manager Resource List	
XMLAuthorizationChecker	Delete Down
XMLEntireXCommandInfoServices	Delete Down Up
PropertySetter	Delete Down Up
XMLEntireXAttachManager	Delete Down Up
EntireXAttachHTTP	Delete Down Up
EntireXAttachMediatorSequence	Delete Down Up
EntireXMediatorDebugger	Delete Down Up
XSLIOHelper	Delete Down Up
XMLCVSInformation	Delete Down Up
XMLEntireXRPCPing	Delete Down Up
PropertyPrinter	Delete Down Up
ActivityListener	Delete Down Up
PackageBuilder	Delete Up
Load new System Manager Resource	

Load System Manager Resource: PackageBuilder	
Class:	PackageBuilder

Save

3 Save the changes.

4 Restart the SAP R/3 Gateway web application. For example, use the Tomcat Manager.

5 After restart, check whether the **System Log** has loaded the `PackageBuilder` resource.

IV

14 Administration and Configuration

The SAP R/3 Gateway provides several tools for administration and configuration to optimize your software environment. This information is provided under the following headings:

15 Using the System Manager

This chapter describes the System Manager you use to control the SAP R/3 Gateway environment. The System Manager is implemented as a web-enabled application and runs as a GUI in your HTML browser. It is described under the following headings:

Technical Components

The main components of the System Manager are

- Stylesheets to generate the HTML output
- Servlet that generates XML

An HTTP request for the System Manager contains optional request data and a stylesheet to transform the output.

Design

The design rules of the graphical frontend were the following:

- Keep it simple.
- Create a pull-down menu to go to all pages as quickly as possible.
- Keep it easy to use, even in critical situations.
- Use no graphics on the HTML pages, thus optimizing performance.
- Design it for administrator and developer alike.

GUI Elements

There are several GUI elements used in the design of the HTML pages.

- Simple hyperlinks for navigation
- Stop

  Command links for creating or changing resources immediately (you will not be prompted to confirm every action). The command link buttons have a grey background.
- HTML forms with a **SAVE** button to change the configuration.

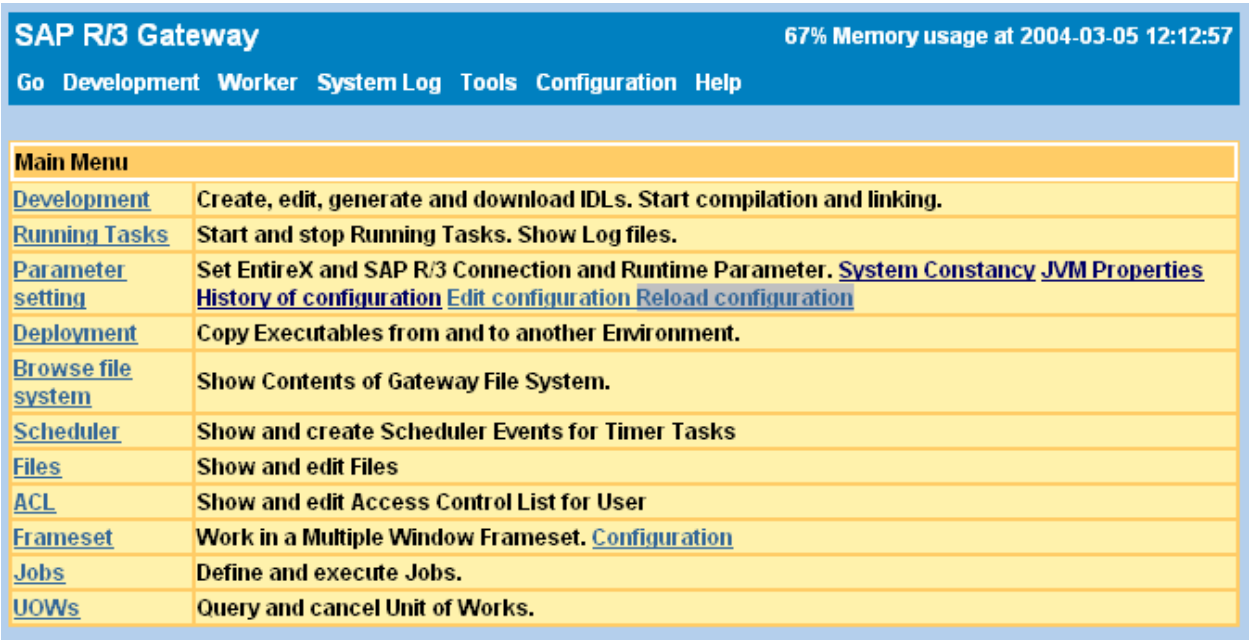
Internal Functionality

The System Manager servlet provides the following functions:

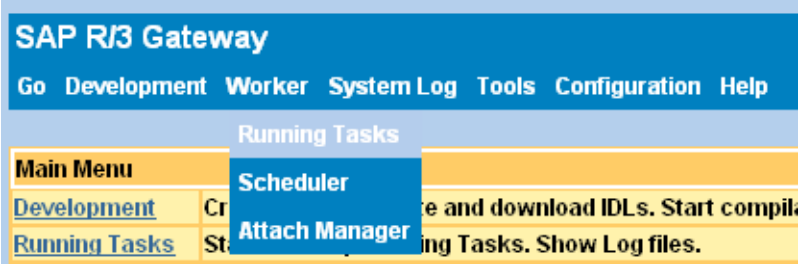
- Start, stop and control running tasks (operating system processes)
- Start jobs (batch or UNIX scripts) and display output
- Built-in scheduling for creating events
- Storage of the configuration in a persistent XML file with history
- List, edit and save files
- Display the **System Log** with a summary of all running tasks
- Support different roles by creating an Access Control List

Usage

You call the starting page with the following URI: *http://YourGateway:8080/sapr3gateway/manager/index*. This displays the main pull-down menu and a short index list.



All menu items become accessible when you move the cursor over them. Some menu items have a submenu, for example:



- Go
- Development
- Worker
- **System Log**
- Tools
- Configuration
- Help

If a new link is available for a page, the background color will change.

System Constants Parameters

A list of the parameters you can set using the System Manager isprovided on the **System Constants** page of the **Configuration** menu or can be accessed using the URI *http://YourGateway:8080/sapr3gateway/manager/SystemConstancy*.

System Constancy		
Title:		SAP R/3 Gateway
System Manager Home Directory:	$SM_HOME	/FS/fs0225/home/thr/tomcat/webapps/systemmanagerRelease
System Manager URL:	SM_URL	http://localhost:8080/systemmanager/EXXR3Gateway
Mailer URL:	MAILER_URL	http://localhost:8080/systemmanager/mailer
EntireX XML-RPC URL:	XML_RPC_URL	http://localhost:8080/exxr3xmlgateway/adapter
Mail to Gateway Administrator:	MAIL_TO_GW_ADMIN	
Script extension:	SCRIPT_EXT	sh
SAG Home Directory:	SAG_HOME	/FS/fs0225/home/thr/64
SAG Environment Script:	SAG_ENV_SCRIPT	SAG_HOME/sagenv.711
EntireX Home Directory:	EXX_HOME	SAG_HOME/exx
EntireX Version Directory:	EXX_VERS	v711
EntireX Bin Directory:	RPC_HOME	EXX_HOME/EXX_VERS/bin
EntireX Library Path:	EXX_PATH	EXX_HOME/EXX_VERS/lib
RFC Library Path:	RFC_PATH	$SM_HOME/RfcSdk.SUN64/lib Windows: $SM_HOME/RfcSdk.NT/lib SunSolaris 32 Bit: $SM_HOME/RfcSdk.SUN32/lib SunSolaris 64 Bit: $SM_HOME/RfcSdk.SUN64/lib AIX 32 Bit: $SM_HOME/RfcSdk.AIX32/lib AIX 64 Bit: $SM_HOME/RfcSdk.AIX64/lib Linux 32 Bit: $SM_HOME/RfcSdk.LINUX/lib Suse z/Linux 32 Bit: $SM_HOME/RfcSdk.SUSEZLINUX32/lib Suse z/Linux 64 Bit: $SM_HOME/RfcSdk.SUSEZLINUX64/lib

The following parameters can be set:

- Title of this application. This is useful if you have more than one instance and you want to differentiate between them.
- Paths and directories to application server, webMethods EntireX and SAP RfcSdk
- Standard development parameters for the RfcIdl tool
- System Log parameters, for example, to change the log level.

The **SAVE** button at the bottom of this page saves the parameters to the XML configuration file. A history file is also maintained. This means that changes can be undone see the section Undo Changes. On this page, there is a parameter called **Configuration Backup number of files**. Use it to set the number of undo operations possible.

You can also use this page to define the constancy to be used in file addresses or URLs. The second column in the table defines placeholder strings which are replaced at runtime. The placeholders are evaluated:

- when starting a task or job.

  Tip: To show the completed replacement, you must set the debug level on the System Manager.

- on URL of timer task
- on List or Edit file
- on List directory contents

With this replacement feature, it is possible to maintain your files or URLs independently of your underlying file system. For example, assume you define `SAG_HOME=/usr/sag`, where `SAG_HOME` is the placeholder. On listing the SAG environment script, you can type in `SAG_HOME/sagenv`. The System Manager calculates the filename as `/usr/sag/sagenv`.

Undo Changes

Every change effected using the **SAVE** button (except file editor changes) is saved in the XML configuration file. A backup file is saved in a subdirectory. You will find the history on the **History of saved Configuration Files** page of the **Configuration** menu or by using the URI *http://YourGateway:8080/sapr3gateway/manager/configHistory*

History of saved Configuration Files	Operation
config.2004-03-05-08-45-47.xml (current active)	Load and Activate Delete
config.2004-03-05-08-12-29.xml	Load and Activate Delete
config.2004-03-05-08-11-57.xml	Load and Activate Delete
config.2004-03-05-08-11-24.xml	Load and Activate Delete
config.2004-03-04-12-54-04.xml	Load and Activate Delete
config.2004-03-04-12-54-01.xml	Load and Activate Delete
config.2004-03-04-12-53-38.xml	Load and Activate Delete
config.2004-03-04-12-48-49.xml	Load and Activate Delete
config.2004-03-04-12-45-48.xml	Load and Activate Delete
config.2004-03-04-12-44-47.xml	Load and Activate Delete
config.2004-03-04-12-40-43.xml	Load and Activate Delete
config.2004-03-04-12-40-32.xml	Load and Activate Delete
config.2004-03-04-12-39-34.xml	Load and Activate Delete
config.2004-03-04-12-38-53.xml	Load and Activate Delete
config.2004-03-04-12-38-24.xml	Load and Activate Delete

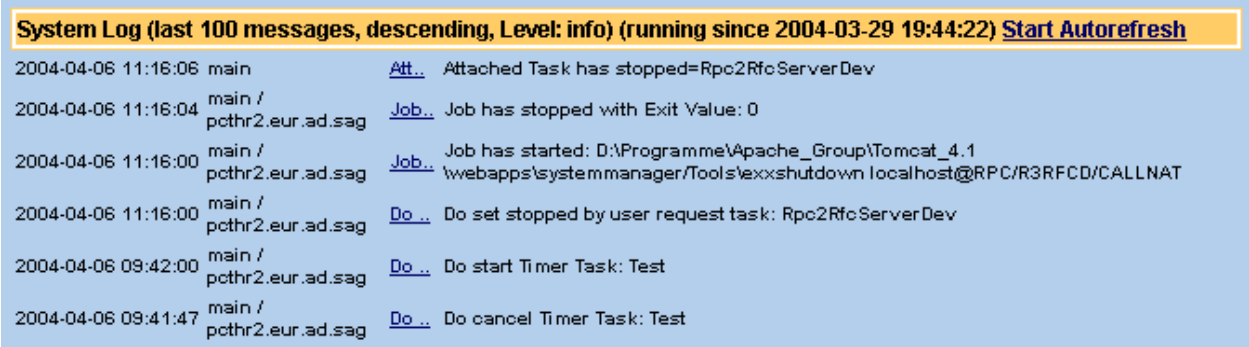
The list is sorted by date and time, the most recent one being listed first. Choose **Load and Activate** to reload a previous version. The following parameters are available to influence the backup history on the **System Constants** page.

Configuration Backup directory	Directory name of saved files.
Configuration Backup History Template	Template to create the backup file.
Configuration Backup Number of Files	Available files after running **Scheduler Timer Task** *CleanUpHistory*

Note: The **Load and Activate** operation does not save the new loaded configuration in the current configuration file. To make the loaded configuration persistent, click Save on **System Constants**.

System Log

The System Manager protocols incoming messages in the System Log. You can display the system log by choosing the **Show**option from the **System Log** menu (or *http://YourGateway:8080/sapr3gateway/manager/systemLog?operation=systemlog*) online and the messages will be written to a file.



System Log (last 100 messages, descending, Level: info) (running since 2004-03-29 19:44:22) Start Autorefresh

2004-04-06 11:16:06	main	Att..	Attached Task has stopped=Rpc2RfcServerDev
2004-04-06 11:16:04	main / pcthr2.eur.ad.sag	Job..	Job has stopped with Exit Value: 0
2004-04-06 11:16:00	main / pcthr2.eur.ad.sag	Job..	Job has started: D:\Programme\Apache_Group\Tomcat_4.1 \webapps\systemmanager/Tools\exxshutdown localhost@RPC/R3RFCD/CALLNAT
2004-04-06 11:16:00	main / pcthr2.eur.ad.sag	Do ..	Do set stopped by user request task: Rpc2RfcServerDev
2004-04-06 09:42:00	main / pcthr2.eur.ad.sag	Do ..	Do start Timer Task: Test
2004-04-06 09:41:47	main / pcthr2.eur.ad.sag	Do ..	Do cancel Timer Task: Test

The online Log has a table with 4 columns.

1. Date and time
2. Message producer ("main" is System Manager, otherwise name of Running Task), followed by "/", user ID or IP-Address of message initiator
3. Short link to view the message text in a new browser window.
4. Message text

All System Log parameters can be changed on the **System Constants** page.

System Manager Log Level:	info Activate
System Manager Log View of max. Message Length:	10000 in bytes. Activate
System Manager Log Home directory:	$SM_HOME/log
System Manager Log Filename Prefix:	EXXR3Gateway
System Manager Log File Life Time:	10 in days. Activate
System Manager Log Limit of cached Messages:	100
System Manager Log Display Sort Order:	descending
System Manager Log auto refresh Time:	5 in seconds.
System Manager Log flush immediately:	true Activate

This table describes the configuration parameters in detail:

System Manager Log Level	Set the level at which the log displays online messages and messages written by a file.
System Manager Log View of max. Message Length	Set the maximum length of messages in bytes which are displayed online. The complete message are always written to a file.
System Manager Log Home Directory:	Home directory of the System Log files
System Manager Log Filename Prefix	Filename prefix of System Log files
System Manager Log File Lifetime	Lifetime of System Log files
System Manager Log Limit of cached Messages	The last number of messages displayed online.
System Manager Log Display Sort Order	Set the sort order. The last incoming message is listed first with sort order `descending` on the System Log page (choose **Show** from the **System Log** menu).
System Manager Log auto Refresh Time	Set automatic refresh time. The browser refreshes the System Log page after this period of time in seconds. You can start the auto refresh on the System Log (choose **Show** from the **System Log** menu).
System Manager Log flush immediately	Set this value to `true`, if each incoming message is to be written to the file immediately. **Tip:** You should set this value to `true` if the log is to be available in realtime for other system management or monitor software.

Some parameters have an **Activate** button. To activate a changed and saved value, you must click this button.

Call from Command Line

Administrators and system-wide monitors like to call the System Manager from the command line. The response should have the content type plain text. The following workshop shows an example of how to get the status of running tasks as text, see **Overview of Running Tasks**.

1. Make an HTTP Get request from (UNIX shell) command line:

```
# Set TOMCAT_HOME to the path of Web Application Server
TOMCAT_HOME=/usr/tomcat; export TOMCAT_HOME
java -cp $TOMCAT_HOME/webapps/sapr3gateway/WEB-INF/lib/SystemManagement.jar HttpGet
```

   Response of HTTP Get Request without URL:

```
usage: Http-Get-Request-URL [ Http-Basic-Auth-UserId Http-Basic-Auth-Password ]
```

2. For the HTTP-Get request URL, you must create a stylesheet to receive the response as text. The following example stylesheet replies with the status of Running Tasks.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">

 <xsl:output method="text" encoding="ISO-8859-1"/>

 <xsl:template match="/">

  <!--
   For each defined Running Task
  -->
  <xsl:for-each select="//configuration[ @name = 'RunningTasks' ]/*">

   <xsl:value-of select="name"/>

   <xsl:variable name="taskname">
    <xsl:value-of select="name"/>
   </xsl:variable>

   <xsl:text>:</xsl:text>

   <!-- Print out status -->
   <xsl:value-of select="//task/status[ ../parameter/name = $taskname ]"/>

   <!-- Print out new line -->
   <xsl:text>&#x0a;</xsl:text>
```

```
  </xsl:for-each>

 </xsl:template>
</xsl:stylesheet>
```

Save this stylesheet as *runStatus.xsl* in the directory *webapps/sapr3gateway*.

3. Make the HTTP-Get request with stylesheet:

```
java -cp $TOMCAT_HOME/webapps/sapr3gateway/WEB-INF/lib/SystemManagement.jar ↩
HttpGet "http://localhost:8080/sapr3gateway/manager/runStatus"
```

The System Manager with stylesheet *runStatus.xsl* replies:

```
Rpc2RfcServerDev:ProcessHasStarted
Rpc2RfcServerInt:
Rpc2RfcServerProd:
Rfc2RpcServerDev:ProcessHasStarted
Rfc2RpcServerInt:
Rfc2RpcServerProd:
PMQServerDev:
```

Tip: Use the *xml.xsl* stylesheet in HTTP-Get requests to obtain all information about the System Manager and running tasks as an XML document.

Change Root Path

The System Manager runs as a servlet in a servlet container. The web applications server tells the servlet in most cases its root installation path. The System Manager uses this to load stylesheets, start kernels and create log files. The **System Constants** page shows the root **Home Directory** path.

In some cases, the **Home Directory** must be changed to your own definition. To do this, you must set the following parameter in file *WEB-INF/web.xml* of all servlets.

Parameter	Description
systemrealpath	Overwrites the default setting from web application server of **Home Directory**. This setting is displayed on the **System Constants** screen. This parameter can also be used to delete spaces in directory path on Windows. (See the section **Using the Setup Wizard**.)
xslpath	Loads stylesheet from an absolute path.
xsl_dir	Loads stylesheet from a relative path. This parameter should be set if the stylesheets are placed in a subdirectory of the root path.
config	Loads this configuration file with path and file name. To load the file with its absolute path, add the protocol `file:/` in front of the parameter. Otherwise the file will be loaded with its relative path.

By default, each servlet loads its own configuration file from the **Home Directory**. The following table shows which file name is used.

Servlet	Default Configuration File Name
SystemManagementServlet	config.xml
TransformerServlet	TransformerConfig.xml
FilterServlet	FilterConfig.xml

When setting a new **Home Directory**, you must define the `config` parameter. Define for each servlet the `config` parameter with the name of the above table and with a leading slash (/) character. Now the configuration file will be loaded from `systemrealpath`.

Note: If you change the *WEB-INF/web.xml* file with the editor, Tomcat reloads the web application immediately.

The following example shows an extract of *web.xml* for setting the parameter:

```
...
    <servlet>
        <servlet-name>&webAppServlet;</servlet-name>
        <servlet-class>SystemManagementServlet</servlet-class>
        <init-param>
            <param-name>xslpath</param-name>
            ↩
<param-value>D:\Programme\Apache_Group\Tomcat_4.1\webapps\sapr3gateway</param-value>
        </init-param>
        <init-param>
            <param-name>systemrealpath</param-name>
            ↩
<param-value>D:\Programme\Apache_Group\Tomcat_4.1\webapps\sapr3gateway</param-value>
        </init-param>
        <init-param>
            <param-name>config</param-name>
            <param-value>/config.xml</param-value>
        </init-param>
     <load-on-startup>1</load-on-startup>
    </servlet>
...
```

Note: The documentation (the sub-directory *docu*) and the *global.js* file cannot be replaced or addressed with this parameter setting. These files must remain in their original locations.

Password Encoding

The System Manager saves the password for several configurations (**Scheduler**, **Attach Manager**) in an XML file. This file and the password contain plain readable text. Since version 2.3.1.07 the password can be saved in a decoded option. The System Manager provides its own algorithms for encoding and decoding. Optionally a password can be saved encoded. If a password cannot decode (because it is not encoded), the original one is passed to the underlying system.

The first time, call the provided page with *http://YourGateway:8080/sapr3gateway/manager/encodePW*. A new menu **Password Encoding** item is created under **Tools**.

After a password is encoded, copy and paste this to the required password fields and save the changed option.

Search Worker

The index and this page provide an input text box to search for configured items. For example, search for `outbound`, you will find all **pipelines**, **Attach Manager**, **Running Tasks** and **Timer Tasks** where this string is contained in the configuration option (title name, queue name, description, ...).

The first time, call the provided page with *http://YourGateway:8080/sapr3gateway/manager/searchWorker*. A new menu **Search Worker** item is created under **Tools**.

16 Workers

The workers are background threads of the **System Manager**, which work asynchronously. The following workers are currently available:

Running Tasks

Running Tasks are long-term processes or daemons. The System Manager can start and control these. The main parts that are controlled are

- handling of standard and error output,
- handling of abnormal termination and
- support of normal termination.

The SAP R/3 Gateway is delivered with several predefined Running Tasks. These are described in the section **Overview of Running Tasks**.

Other pages are supported by the function Configuration of all Running Tasks; choose **Parameter** from the **Configuration** menu (or *http://YourGateway:8080/sapr3gateway/manager/parameter*) :

Rpc2RfcServerDev	Running Task Environment Copy to Delete Down
Rpc2RfcServerInt	Running Task Environment Copy to Delete Down Up
Rpc2RfcServerProd	Running Task Environment Copy to Delete Down Up
Rfc2RpcServerDev	Running Task Environment Copy to Delete Down Up
Rfc2RpcServerInt	Running Task Environment Copy to Delete Down Up
Rfc2RpcServerProd	Running Task Environment Copy to Delete Down Up
Rfc2RpcServerThrDev	Running Task Environment Copy to Delete Down Up
DCOMServer	Running Task Environment Copy to Delete Down Up
PMQServerDev	Running Task Environment Copy to Delete Up
Create new Running Task Environment	

The following table describes the links and the command links:

Running Task Environment	Go to the configuration of the selected running task.
Copy to	Copy all configuration settings of the one selected to a new one. The new running task has the old name with prefix `Copy-Of`. It is appended to the end of the list. **Note:** The external configuration file of a **Rpc2Rfc kernel** is not copied. Copy and rename an existing RPC server configuration file in file system and set the new filename as parameter in the new one.
Delete	Delete the configuration of the selected running task. It cannot be deleted if it is already running.
Down	Set this after the next one.
Up	Set this before the previous one.

Create new Running Task Environment	Create a new empty running task. You will get a dialog which asks for a name and a type.

To create a new running task, choose one of the following types. The type later defines which environment parameters will be available.

Rpc2Rfc	Environment for an **Rpc2Rfc kernel**
Rfc2Rpc	Environment for an **Rfc2Rfc kernel**
Java (PMQ) Server	Environment for a Java server process.
Natural RPC Server	Environment for a Natural RPC Server
Starts server with shell script	Environment for any shell script that starts a controlled process with `exec`.

For each running task, you must define the command to start the process. The process is created as a child process of Application Server with

- `fork() and exec()` on UNIX or
- `CreateProcess()` on Windows.

The System Manager can only control the progress of its next child. It is not possible to control a process which was started as the child process of a shell script (the child of a child process). In this case, use the `exec` shell script command to replace the parent and give process control to the System Manager.

The standard and error output are redirected to a file. The name of this file consists of:

1. Current directory path
2. Kernel task name
3. A timestamp
4. The extension ".log"

A new file is evaluated by the **Scheduler** in Timer Task NewLog. If a running task is started or stopped, the **Log** command button is available, see **Overview of Running Tasks**. This command will display the output content on the next page.

D:\Programme\Apache_Group\Tomcat_4.1\webapps\systemmanager/Rpc2Rfc/run/dev\Rpc2RfcServerDev.2004-04-05.log

```
Server Parameters:
Broker Id (default)................. ..localhost:1971:TCP
Service   (default)................. ..RPC/R3RFCD/CALLNAT
MinWorker (free for new conversation) 1
MaxWorker (max parallel active) ..... 1
EndWorker (criteria when to stop) ... N
Timeout   (in seconds for Broker) ... 30
Api used  (for Broker) ............... AUTO
Codepage  (for data conversion) ..... Non-ECS
Server is starting.................... Mon Apr 05 14:16:53 2004
Broker available, server continues....Mon Apr 05 14:16:54 2004
```

Tip: Before creating a running task of your own, consider copying an existing one.

Tip: Define the environment and parameter with the placeholder and replacement definition on the **System Constants** page.

Tip: Set the log level to `debug`, if you want to display your environment and parameter definition when a running task is started, see **System Log**.

Scheduler

The scheduler is a collection of timer tasks. A timer task has a time and/or a period of time. If this time is scheduled, an HTTP Get request with a specific URL will be executed.

Timer Task	Operation	Startup	At	Period	Next At	Result
CleanUpHistory	Stop Simulate		00:10:00 05-05	1 Day(s) 00:00:00	00:10:00 05-06	Show Text
CleanUpLog	Stop Simulate		00:05:00 05-05	1 Day(s) 00:00:00	00:05:00 05-06	Show Text
NewLog	Stop Simulate		00:00:00 05-05	1 Day(s) 00:00:00	00:00:00 05-06	Show Text

Create persistent Timer Task

Create new temporary Timer Task

Delete Exceptions

Edit Timer Task:	
Name:	
Startup:	auto
Start at:	empty (in one minute), HH:mm or yyyy-MM-dd HH:mm or EEE HH:mm
Period:	optional, repeat time in seconds
At fixed Rate:	true
Write Response to System Log:	false
Command:	required, URL of HTTP-Get-Request
User ID:	optional, if HTTP-Basic-Authentication is needed.
Password:	optional, if HTTP-Basic-Authentication is needed.

Save

There are several predefined timer tasks.

Name	Description	At	Period	URL
NewLog	Set new log file for System Manager Log and all running tasks	00:00	86400 sec. or 24h	SM_URL/timerTaskResponse?operation=setnewlog
CleanUpLog	Delete old log files whose lifetime has expired. The lifetime is configured on the**System Constants** page. See	00:05	86400 sec. or 24h	SM_URL/cleanup

Name	Description	At	Period	URL
	Clean up Log Files.			
CleanUpHistory	Delete old System Manager configuration files. See **Undo Changes**.	00:10	86400 sec. or 24h	SM_URL/configHistory

It is possible to create your own timer task. To do this you must decide whether you want to create a temporary or a persistent task. Tempory timer tasks are not stored in the configuration file. The definitions will be lost when the System Manager is restarted. The definition of a persistent timer task is saved in the configuration file. A persistent timer task also has the attribute Startup. The Startup type `auto` starts the timer when the system is started. With `manual`, you can start the timer task later. The following parameters are available for a timer task:

Name	Name of timer task.
Startup	For persistent timer tasks only: `auto` creates this timer task when the System Manager is started.
Start at	Defines the starting time.
Period	Defines the interval in seconds.
At fixed rate	In fixed-rate execution, each execution is scheduled relative to the scheduled execution time of the initial execution.
Write Response to System Log	`true` writes the response of the scheduled request to the **System Log**.
Command	HTTP Get request URL.
User ID	HTTP Basic Authentication user ID.
Password	HTTP Basic Authentication password.

Depending on its state, several operations are available to a timer task.

Start	The persistent timer task can start.
Stop	The timer task is stopped. No more scheduled events will be created.
Simulate	If you do not want to wait until Start-at time, you can simulate the scheduled event. This operation makes a test of all parameters.
Delete	Delete timer tasks that have not yet started. If the timer task is persistent, it will also be deleted from the configuration file.

After running a scheduled event, the output will be available with the following links:

Show	Displays the results in a new browser window.
Text	Displays the results at the end of the next page as text.
Exception	This link will be available if an error occurs while an event is running. The message is displayed at the end of the next page.

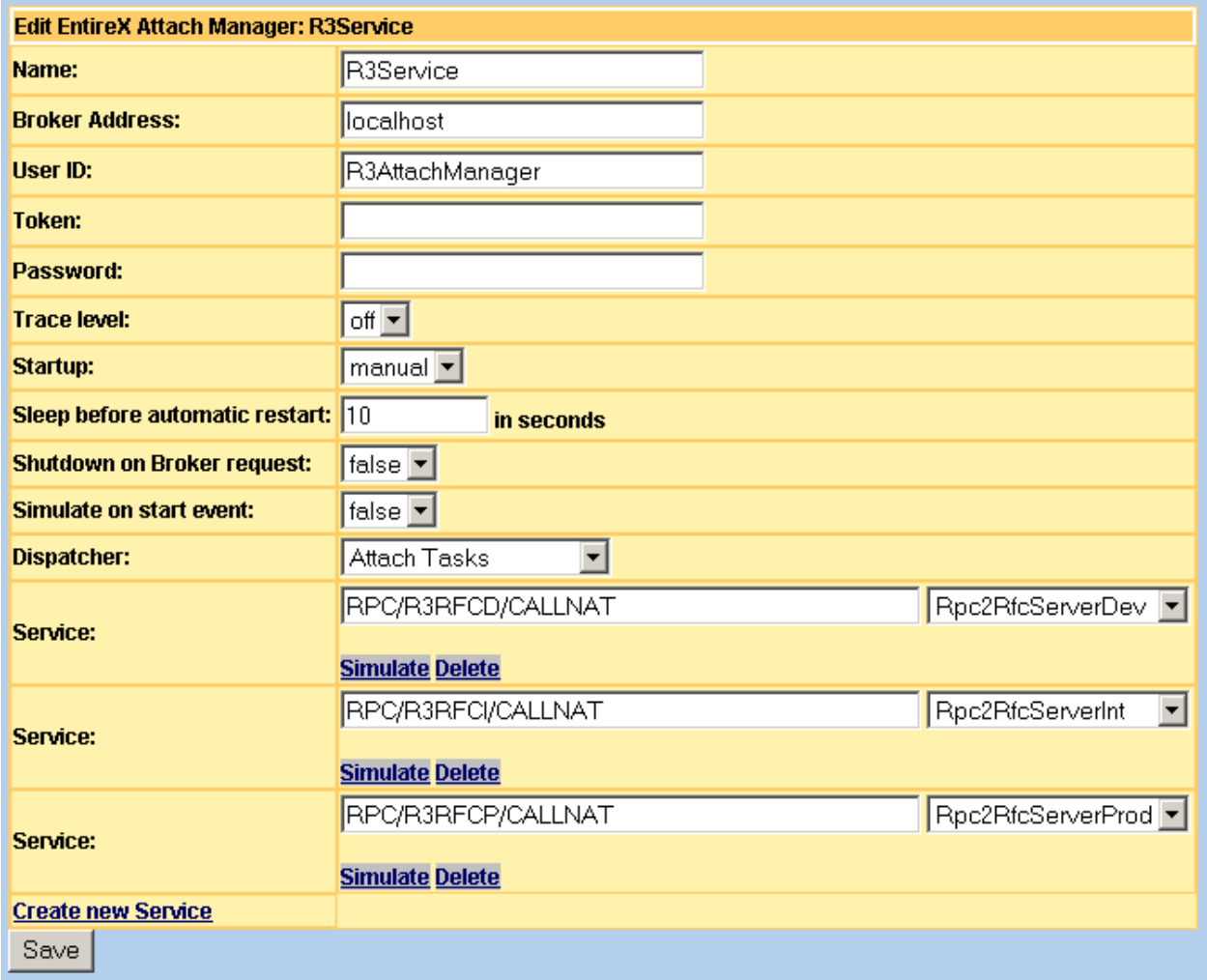
Attach Manager

The webMethods EntireX Attach Manager in this System Manager is implemented as an asynchronous thread, which is informed by the connected webMethods EntireX Broker if a server is missing. For example, a client sends a message to the webMethods EntireX Broker. There is no server connected to the Broker, but there is a registered Attach Manager. The Attach Manager thread receives the event and implements several dispatchers to forward this event.

Dispatcher	Description
Attach Tasks	The missing server event will be forwarded to start a configured running task, see **Overview of Running Tasks**.
HTTP Request	The event will be forwarded to make a specific HTTP-Get request. The broker ID and service are passed as HTTP-Get variables `broker` and `service`. The Attach Manager thread waits until the HTTP-Get command proceeds. When the HTTP-Get command is finished, the Attach Manager performs a wait-for-receive command on the EntireX Broker to retrieve the next event.
Mediator Sequences	The event will be forwarded to make a specific HTTP-Get request. The specific URL will start a Service Orchestrator (Mediator) sequence. The BrokerID, service, user ID and password are passed as HTTP-Get variables `xbd.pmq.broker`, `xbd.pmq.service`, `xbd.pmq.userid` and `xbd.pmq.password`.

Depending on its state, several operations are available to an Attach Manager.

Start	The Attach Manager can start.
Stop	The Attach Manager is stopped.
Simulate	If you do not want to wait, you can simulate the attach event. This operation makes a test of all parameters. This operation works only when the Attach Manager has been started once.
Delete	Delete Attach Managers that have not yet started.
Restart	Stop and start Attach Manager.
Copy to	Copy all parameters to a new instance.



Each Attach Manager has the following parameters:

Name	The Attach Manager's name in the System Manager.
Broker address	IP or DNS Name of webMethods EntireX Broker.
User ID	User ID registered with webMethods EntireX Broker .
Token	Registered Token.
Password	User ID's password with webMethods EntireX Security.
Trace level	Trace calls with level 1,2,3 or switched off.
Startup	`auto` starts the Attach Manager when the System Manager is started.
Sleep before automatic restart	Wait this period of time between termination and the next start.
Shutdown on broker request	`true` terminates the Attach Manager if a shutdown event is received. The Value `false` will restart it.
Simulate on start event	When the Attach Manager is started, each service will get a missing server event. Each service will then look in its queue and process any messages that have been received in the meantime.

Dispatcher	Select the dispatcher.
Service	List of attached services.

Each service has the following operation or parameters:

Queue	The queue name: CLASS/SERVER/SERVICE
Simulate	This service gets a missing server event to process the message in its queue.
Delete	Delete this service.

The dispatcher's `HTTP Requests` and `Mediator Sequences` have additional parameters:

URL	Perform this HTTP-Get request if a server event is missing.
Log response	`true` writes the output of an attached event to the System Log.
Sleep interval between 2 events	Wait this period of time between 2 processed events.
Synchronize parallel requests	■ The value `false` does not synchronize between parallel requests. The stylesheet called must be able to handle parallel processing. ■ The value `true` synchronizes between all services of attach managers with this value. If a request is running, all others wait. ■ The value `this service` is only synchronized for parallel processing. Another request for this service will wait until the first has finished. This value is default and recommended. ■ The value `this service without waiting` is only synchronized for parallel processing. Another request for this service is aborted because the first is working. An error message is placed into the **System Log**.

17 Optimization Tools

This section contains a collection of tools and wizards to optimize and control your distributed system components.

Ping Wizard

The Ping Wizard is a utility to perform all of the steps for controlling your production environment. Each step is explained on the **Ping Wizard** page of the **Tools** menu (or *http://YourGateway:8080/sapr3gateway/manager/pingWizard*).

webMethods EntireX Broker High Water Mark

This tool allows you to control and check the webMethods EntireX Broker High Water Marks (choose **Broker High Water Marks** from the **Tools** menu or *http://YourGateway:8080/sapr3gateway/manager/brokerHWM*). When this page is called, all listed webMethods EntireX Brokers are queried for their high water marks. The results are shown in percent usage.

Broker: ibm1:3800			
Short Buffer high	1135	7204	15% usage
Short Buffer active	22	7204	0% usage
Long Buffer high	24	2004	1% usage
Long Buffer active	2	2004	0% usage
Server high	45	200	22% usage
Server active	37	200	18% usage
Service active	26	100	26% usage
Conversation high	640	2002	31% usage
Unit of Work active	0	10	0% usage

Broker List	
localhost:1971	Delete Down
ibm1:3800	Delete Up
New Broker	
Send Mail, if High Water Mark is reached	

Edit Broker High Water Mark Configuration: ibm1:3800	
Address:	ibm1:3800
User ID:	
Password:	
Warning Limit on:	in percent. Default is 90%.

Save

You can define a warning limit in percent for each broker. If one of the Broker resources reaches this limit, a message will be sent to the **System Log**. An e-mail will be generated if this page is called with `mail=` parameter. The e-mail address is configured on the **System Constants** page. The IP-Address or DNS name of the SMTP server is set with **JVM Properties**.

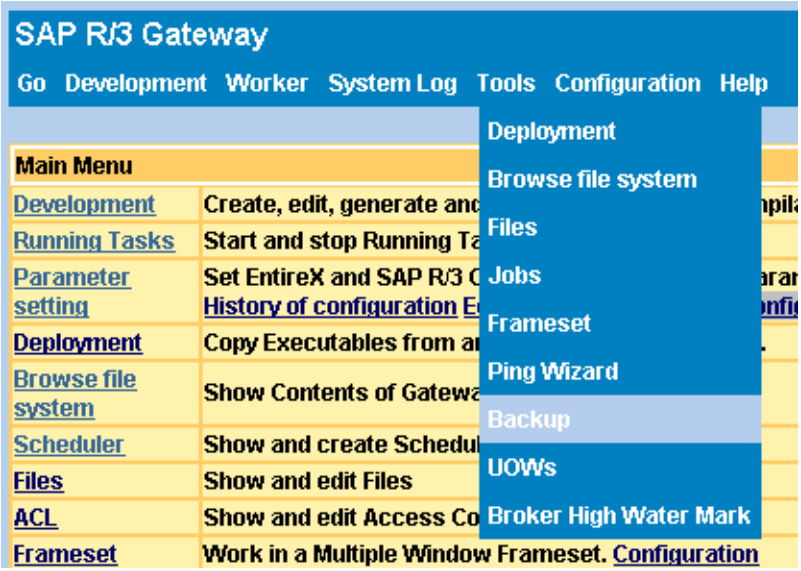
Note: The High Water Mark limit will only work successfully, if no limitation is defined for a service. The best way to do this is to share all broker resources with all services. You can detect this kind of resource bottleneck for a service in the Broker log file.

Tip: Create a **Scheduler Timer Task** with command URL *SM_URL/brokerHWM?mail=* to control and check the webMethods EntireX Broker.

Backup

The backup tool collects all files of *webapps/sapr3gateway* and subdirectory and creates a ZIP file. You can use this ZIP file later to extract saved files or to use as WAR file to build a second instance of SAP R/3 Gateway.

The backup script gets the current timestamp as parameter. The resulting ZIP file has this timestamp as postfix. If you start the backup tool from the System Manager menu, the timestamp is evaluated with the current day's date.



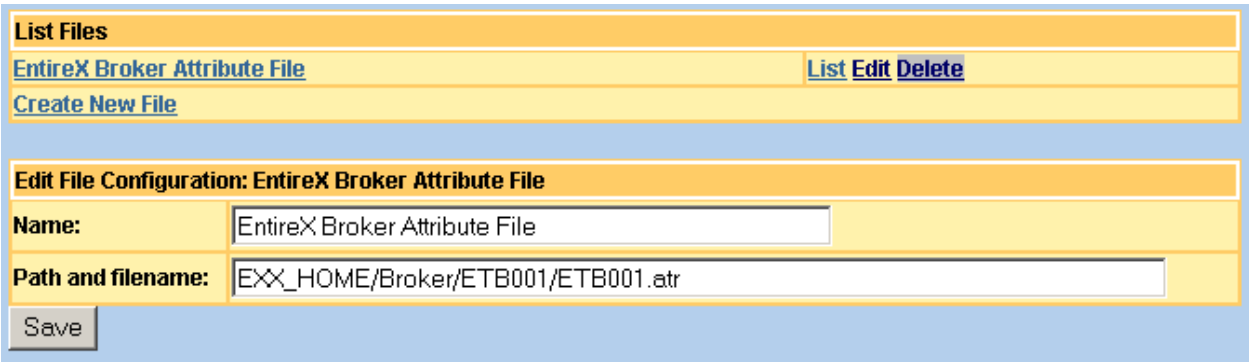
Depending on your operating system and the number and size of files, the backup script may take a lot of time. The System Manager on the **System Constants** page of the **Configuration** menu has a **Job Controller Timeout** value, which determines how long a job can run. If this timeout is reached, the System Manager will kill the backup script. Therefore at first backup time, you should call the backup script *backup.bat* on Windows or *backup.sh* on UNIX with the command line utility in directory *webapps/sapr3gateway* and check the elapsed time.

Files

The **Files** tool from the **Tools** menu allows you to optimize your work with files, for example configuration files. The following commands are available:

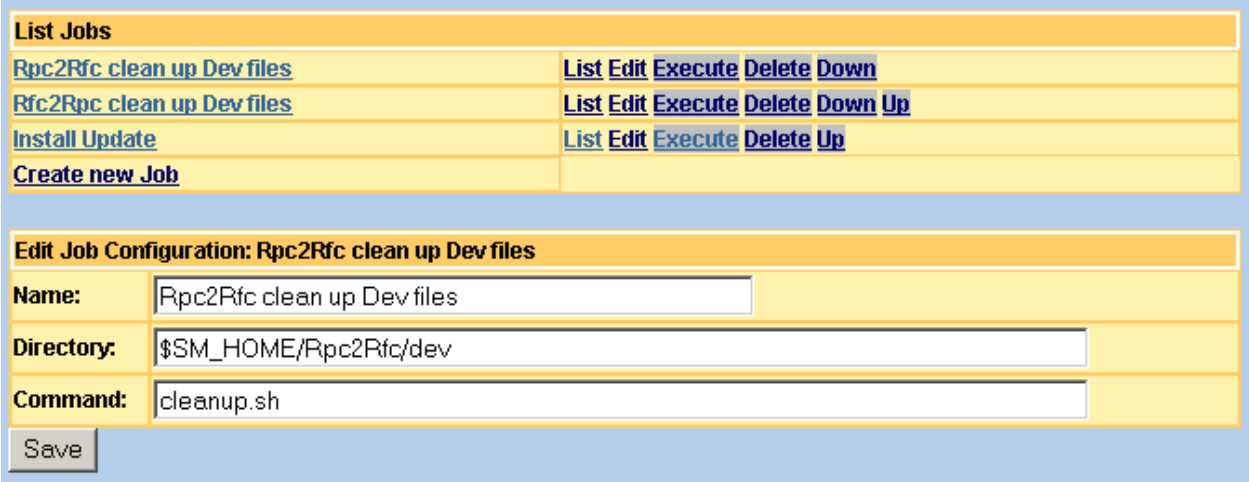
List	Lists the contents of a file at the end of the next page.
Edit	Starts a simple text editor at the end of the next page.
Tail	Only on UNIX: Lists the last 100 lines of the file. **Tip:** This feature is helpful if you have big files and you only want to see the end of a file. You can change the tail command (number of lines) after one execution call in the address line of the browser.
Delete	Deletes the item from the list.
Up	Pushes up this item in the list.
Down	Pushes down this item in the list.
Create New File	Creates a new item in the list.

The following example shows the webMethods EntireX Broker Attribute file as an item:



Jobs

The **Jobs** tool from the **Tools** menu allows you to maintain a list of your shell scripts on UNIX or batch files on Windows. You can edit and execute this.



The following commands are available:

List	Lists the shell script or batch file
Edit	Starts an editor with shell script or batch file
Execute	Starts this shell script or batch file as job and the next page shows the output result with response code.
Delete	Deletes the item from the list
Up	Pushes up this item in the list
Down	Pushes down this item in the list

The System Manager on the **System Constants** page of the **Configuration** menu has a **Job Controller Timeout** value to determine how long a job can run. If this timeout is reached, the System Manager will kill the job. Instead of the output result, you will get the error message `Process is destroyed because time limit exceeded`.

The UNIX installation delivers two predefined jobs. Each shell script deletes files in the **Development**, which are created during **Compile And Link**. This helps if you want to generate all files after a new installation of webMethods EntireX for example.

Frameset

This link starts your Frameset definition, which you define in Frameset configuration. A Frameset helps you to handle multiple installations of the SAP R/3 Gateway web application in one browser window. The following example shows the

1. Default instance,
2. Documentation and
3. System log of the Hamburg Linux instance.

SAP R/3 Gateway - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück Suchen Favoriten Verlauf

Adresse http://suntb01:8080/sapr3gateway/manager/frame Wechseln zu Links

SAP R/3 Gateway 82% Memory usage at 2004-04-07 14:50:26

Go Development Worker System Log Tools Configuration Help

Main Menu	
Development	Create, edit, generate and download IDLs. Start compilation and linking.
Running Tasks	Start and stop Running Tasks. Show Log files.
Parameter setting	Set EntireX and SAP R/3 Connection and Runtime Parameter. System Constancy JVM Properties History of configuration Edit configuration Reload configuration
Deployment	Copy Executables from and to another Environment.
Browse file system	Show Contents of Gateway File System.
Scheduler	Show and create Scheduler Events for Timer Tasks

EntireX Communicator

SAP R/3 Gateway | Version 2.1.1

HOME UP PREV NEXT

Documentation Overview

Installation and Getting Started
Release Notes
Introducing SAP R/3 Gateway
Installation and Configuration

Reference
EntireX Communicator
Tomcat Documentation

Programming

Administration and Tools

HAMLinux EntireX SAP R/3 Gateway 93% memory usage at 2004-04-07 15:03:05

Go Development Worker System Log Tools Configuration Help

System Log (last 100 messages, descending, Level: info) (running since 2004-01-27 14:12:11) Start Autorefresh

2004-04-07 01:05:01	main / localhost	Fil..	File is deleted: /opt/jakarta/tomcat/webapps/systemmanager/log/EXXR3Gateway.2004-03-27.log
2004-04-07 01:00:00	main / localhost	Set..	Set new logs for running tasks
2004-04-07 01:00:00	main / localhost	Do ..	Do set System output log to: /opt/jakarta/tomcat/webapps/systemmanager/log/EXXR3Gateway.2004-04-07.log
2004-04-06 01:05:02	main / localhost	Fil..	File is deleted: /opt/jakarta/tomcat/webapps/systemmanager/log/EXXR3Gateway.2004-03-26.log

Unbekannte Zone (Gemischt)

Clean up Log Files

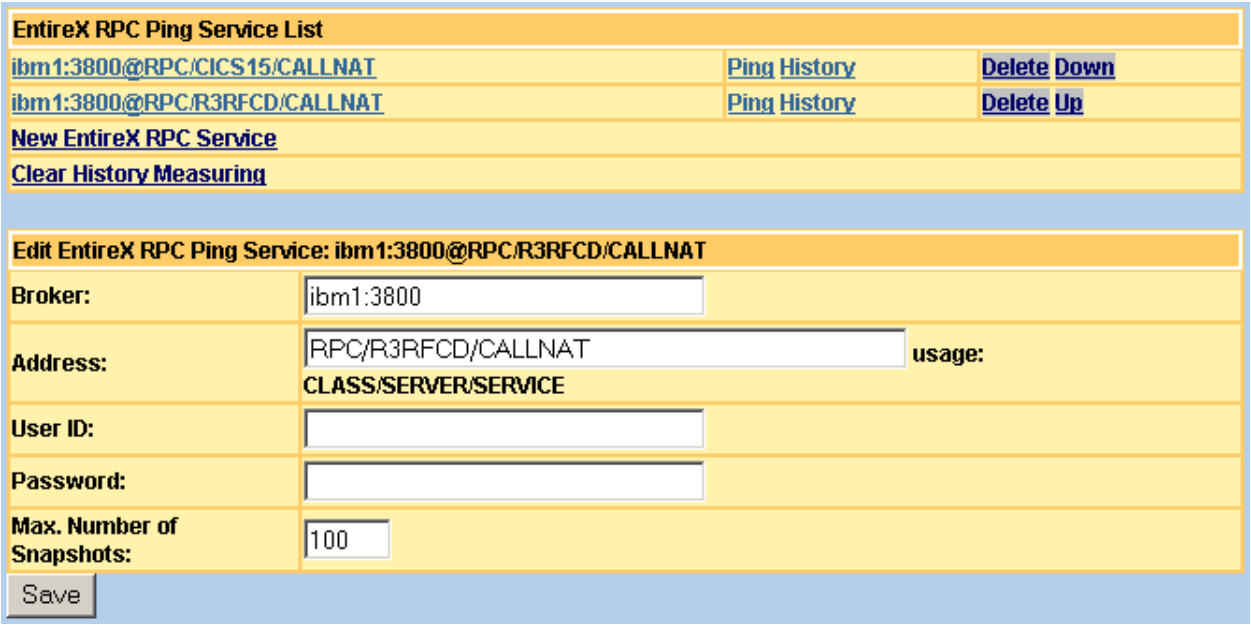
Every night, a new log file is set by the **Scheduler Timer Task** *NewLog*. Old files can be deleted. The lifetime of log files is defined in **System Manager** or for each **Running Task**. The request *http://YourGateway:8080/sapr3gateway/manager/cleanup* shows the following output for example.

Clean up Log Files			
$SM_HOME/Rpc2Rfc/run/dev/*Rpc2RfcServerDev*	Last Modified	Size	Status
Rpc2RfcServerDevMF034.2004-06-20.log	2004-06-20 00:00:00	0	Delete
Rpc2RfcServerDev.2004-06-20.log	2004-06-20 23:59:33	145103	
Rpc2RfcServerDevMF034.2004-06-21.log	2004-06-21 15:17:52	8409	
Rpc2RfcServerDev.2004-06-30.log	2004-06-30 00:00:00	0	
Rpc2RfcServerDev.2004-06-19.log	2004-06-19 23:59:30	145103	Delete
Rpc2RfcServerDev.2004-06-21.log	2004-06-21 16:17:43	89104	
Rpc2RfcServerDevMF034.2004-06-22.log	2004-06-22 16:29:37	31205	
Rpc2RfcServerDev.2004-06-22.log	2004-06-22 17:28:19	116814	
Rpc2RfcServerDevMF034.2004-06-23.log	2004-06-23 19:27:57	1284	
Rpc2RfcServerDevMF034.2004-06-30.log	2004-06-30 00:00:00	0	
Rpc2RfcServerDev.2004-06-23.log	2004-06-23 00:00:00	0	
Rpc2RfcServerDevMF034.2004-06-24.log	2004-06-24 00:00:00	0	
Rpc2RfcServerDev.2004-06-24.log	2004-06-24 00:00:00	0	
Rpc2RfcServerDevMF034.2004-06-25.log	2004-06-25 00:00:00	0	
Rpc2RfcServerDev.2004-06-25.log	2004-06-25 00:00:00	0	
Rpc2RfcServerDevMF034.2004-06-26.log	2004-06-26 00:00:00	0	
Rpc2RfcServerDev.2004-06-26.log	2004-06-26 00:00:00	0	
Rpc2RfcServerDevMF034.2004-06-27.log	2004-06-27 00:00:00	0	
Rpc2RfcServerDev.2004-06-27.log	2004-06-27 00:00:00	0	
Rpc2RfcServerDevMF034.2004-06-28.log	2004-06-28 00:00:00	0	
Rpc2RfcServerDev.2004-06-28.log	2004-06-28 00:00:00	0	
Rpc2RfcServerDevMF034.2004-06-29.log	2004-06-29 00:00:00	0	
Rpc2RfcServerDev.2004-06-29.log	2004-06-29 00:00:00	0	
$SM_HOME/Rpc2Rfc/run/dev/*Rpc2RfcServerDevMF509*	Last Modified	Size	Status

The log file's timestamp is evaluated against the defined lifetime and depending on the result, the file is deleted. By default, this tool is called every night by the **Scheduler Timer Task** *CleanUpLog*

RPC Ping

In a distributed environment, where the webMethods EntireX Broker and webMethods EntireX RPC Server reside on different platforms, you can measure the elapsed time of an RPC ping call. The **RPC Ping** page of the **Tools** menu (*http://YourGateway:8080/sapr3gateway/manager/rpcPing*) shows the list of available RPC Services and how to create your own.



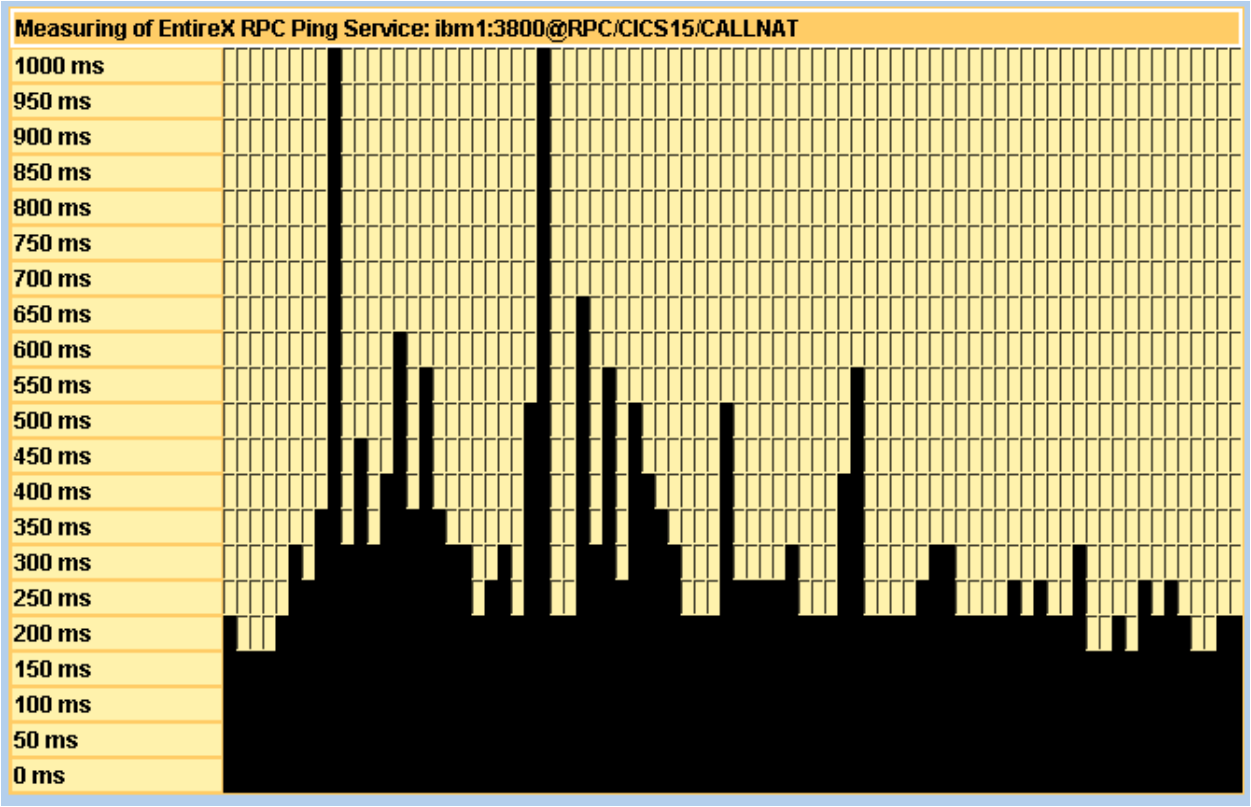
What does the RPC ping measure? This web application creates an webMethods EntireX RPC call with ping option to the webMethods EntireX Broker. The Broker dispatches this request to the RPC server and the response goes to the client. The response time of your network traffic, webMethods EntireX Broker and available RPC server is measured. This feature allows you to optimize your RPC server environment (start new replicates for example) or detect other bottlenecks.

The following operations are available:

New webMethods EntireX RPC Server	Open a dialog to create a new service.
Clear History Measuring	Clear all measuring points.
Ping	Ping and measure the elapsed time of this service. The elapsed time is remembered. A version message from server is displayed.
History	Shows the history of measuring points of this service.
Create Time Task	Open a dialog to create a **Scheduler Timer Task** with a given parameter.
Delete	Delete a service.

Up	Change the position of this service.
Down	Change the position of this service.

For example, the history of measuring points is displayed as a diagram:



To create your own service for a **new EntireX RPC server**, you must set the following parameters:

Broker	IP/DNS and port address of webMethods EntireX Broker
Address	Service name CLASS/SERVER/SERVICE
User ID	Set the webMethods EntireX client user ID.
Password	Set the webMethods EntireX security client password.
Max. Number of Snapshots	Number of remembered measuring points. If this number is reached, the first will be deleted.

Create a **Scheduler Time Task** to measure a service in a period of time. Use the following HTTP Get request command:

```
SM_URL/rpcPing?operation=exxrpcPing&broker=YourBroker&service=CLASS/SERVER/SERVICE&snapshots=100
```

Note: If no network, webMethods EntireX Broker or RPC server is available, no measuring point will be created.

Event Dispatcher

The Event Dispatcher listens in the **System Log** and checks all messages to determine if they have a match in an event list. If a message is matched, an event will be created. The event is performed as an HTTP Post request. It is possible to call a stylesheet to consume the created event. The stylesheet *mailEventDispatcher* consumes the events and creates an e-mail.

How to install

On the first request of the stylesheet *eventDispatcherAdmin.xsl* with URL *http://YourGateway:8080/sapr3gateway/manager/eventDispatcherAdmin*, the Event Dispatcher is registered in the **System Manager** and a menu item is created under the **System Log** top-level menu. For the installation to be completed and to take effect, a restart of the SAP R/3 Gateway web application is required.

How to uninstall

To uninstall the Event Dispatcher, call the resource administrator stylesheet with URL *http://YourGateway:8080/sapr3gateway/manager/resourceAdmin*, delete the class `EventDispatcher` and restart the web application. To remove the menu item, call the menu administrator stylesheet with URL *http://YourGateway:8080/sapr3gateway/manager/menuAdmin*.

How to configure the Event Dispatcher

The Event Dispatcher dispatches the event with an HTTP Post request.

Parameter	Description
URL	Define an HTTP Post endpoint URL. You can use the **System Constants Parameter**. For example, `SM_URL/mailEventDispatcher` sends the event to itself and calls the stylesheet *mailEventDispatcher.xsl*.
User ID	HTTP Basic Authentication user ID.
Password	HTTP Basic Authentication user password.

How to configure the e-mail notification

To send e-mails, the SMTP server must be set. Use the **JVM Properties** page to set the `mail.smtp.host`.

The stylesheet *mailEventDispatcher.xsl* sends an e-mail with the **XSLT extension**. The following parameters are requests from **System Constants Parameter**.

Parameter	Description
Title	The title of SAP R/3 Gateway is used for the subject of the e-mail.
MAIL_TO_GW_ADMIN	Set a list of receiver e-mail addresses separate by blank spaces.
MAIL_FROM_GW_ADMIN	Set the e-mail sender address. For example, use the host name.

How to create events for notification

Click on the link **Create Notification Event** and set the following parameters.

Parameter	Description
Name	Define a name for your event. The name is displayed and can be used in the notification e-mail to identify this event definition. Set a name for the reason why this message (or resource) pattern is matched.
Message Pattern	Define a string as pattern for the matching message line. Use * as wild card definition at the beginning or the end of the message line.
Resource Pattern	Define a string as pattern for the matching message line. Use * as wild card definition at the beginning or the end of a resource name.

The following message pattern can be used to create events when starting or stopping SAP R/3 Gateway.

Message Pattern	Resource Pattern	Event
System manager is started	main	The **System Manager** has finished the initialization and has started completely.
Shutdown system manager	main	The shutdown process is starting. All resources (e.g. Running Tasks, Timer Tasks, ...) will close and stop.

How to create your own event consumer

You can create your own stylesheet to consume the events. Use *mailEventDispatcher.xsl* as a template and copy this to your own file. Then change the HTTP Post URL to call the new stylesheet.

18 Configuration of System Manager

There are several pages available to help you to configure the SAP R/3 Gateway web application.

System-wide Parameters

The global system-wide configuration parameters are described on the page **System Constants**.

Access Control List

The access control list can control each request for a user. The following screen snapshot shows the default configuration.

User list	Has assigned Rule(s)	Operation
anonymous	Administrator	Delete
Create User		

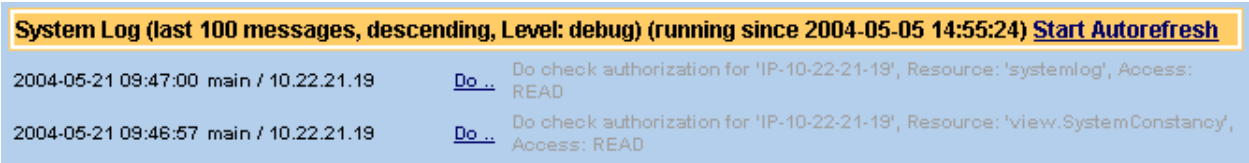
Role list	Has assigned Resource(s)	Operation
Administrator	anyone (modify), anyone (delete)	Delete
DeveloperRf2Rpc	view.index (read), view.dev (read), view.run (read), view.dep (read), task.Rfc2RpcServerDev (modify), file.$SM_HOME/Rfc2Rpc/dev/RFCRPC.idl (modify), file.$SM_HOME/Rfc2Rpc/dev/PMQ.idl (modify), job.$SM_HOME/Rfc2Rpc/dev/make.sh* (modify), task.Controller (modify), file.$SM_HOME/Rpc2Rfc/run/dev/Rfc2RpcServerDev* (read), job.$SM_HOME/Rfc2Rpc/run/dev/deployFrom.sh (modify), task.Rfc2RpcServerDev (modify)	Delete
DeveloperRpc2Rfc	view.index (read), view.dev (read), view.run (read), view.dep (read), file.$SM_HOME/Rpc2Rfc/dev/R3RFC.idl (modify), job.$SM_HOME/Rpc2Rfc/dev/make.sh* (modify), task.Controller (modify), job.$SM_HOME/Tools/exxshutdown.sh localhost@RPC/R3RFCD/CALLNAT (modify), file.$SM_HOME/Rpc2Rfc/run/dev/Rpc2RfcServerDev* (read), job.$SM_HOME/RfcIdl/rfcidl.sh* (modify), job.$SM_HOME/Rpc2Rfc/run/dev/deployFrom.sh (modify), task.Rpc2RfcServerDev (modify)	Delete
Create Role		

How is the default access control list read? Each request has a caller. This caller is searched for in the user list. If this caller is not found, *anonymous* is used. Each user (*anonymous*) has a role with several resources (*anyone*). The accessed resource of the request is checked in the request list of the assigned role. *anyone* is a placeholder for all resources.

Each request is logged in `debug` mode. The **System Log** displays

- identity information about user ID or IP/DNS address,

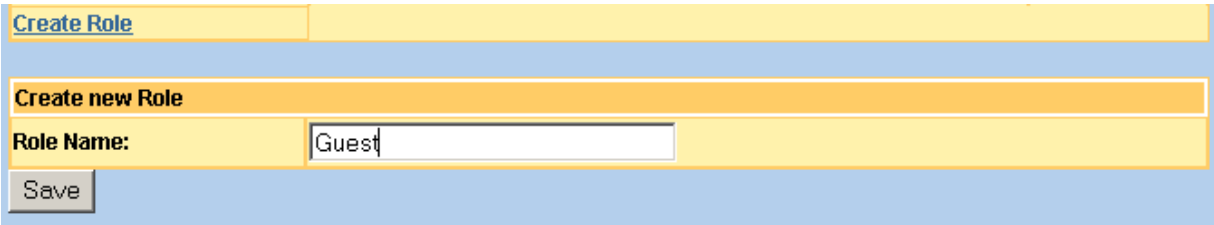
- accessed resource and
- access mode (`READ`. `MODIFY` or `DELETE`).



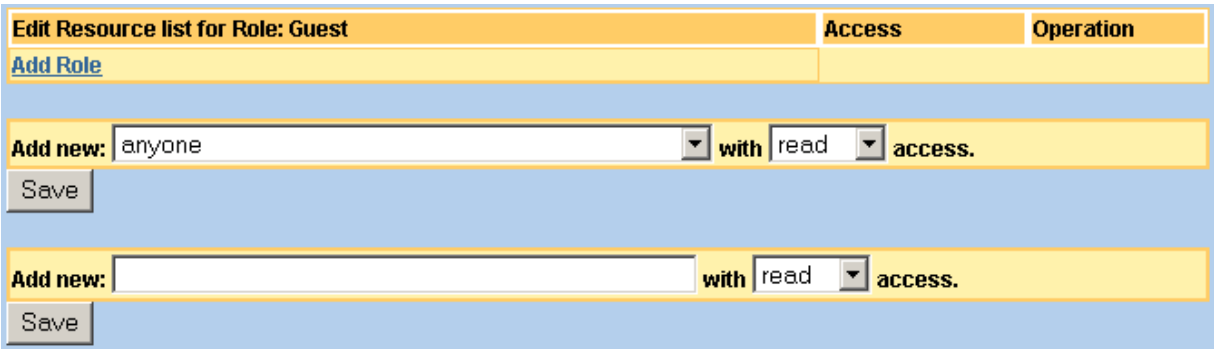
A user request is identified is by user ID and/or IP/DNS address. You will see the user ID only if this web application is configured with security. Use the **Setup Wizard** on the **Help** menu to switch on security. If you want to identify the requests by IP/DNS address, use the following naming convention: `IP-#1-#2#-3-#4`. *#1* up to *#4* are decimal numbers of the IP address. All dot characters are replaced with a hyphen. Examples: `IP-127-0-0-1`, `IP-pcFrankfurt-de` or `IP-localhost`.

By default, there is a role *DeveloperRf2Rpc* and *DeveloperRpc2Rfc* assigns this role to a user. This user can only perform development steps. The following little workshop explains how to create a guest role for users.

1. Create and save a role *Guest*

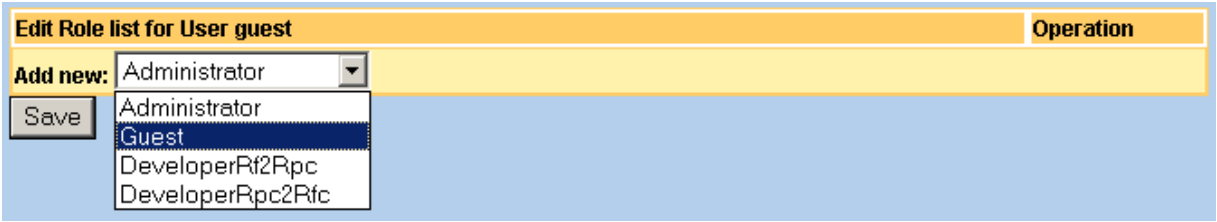


2. Press new *Guest* link in role list.
3. Press **Add Resource** in empty resource list



4. Press **Save** to add *anyone* with *read* access.

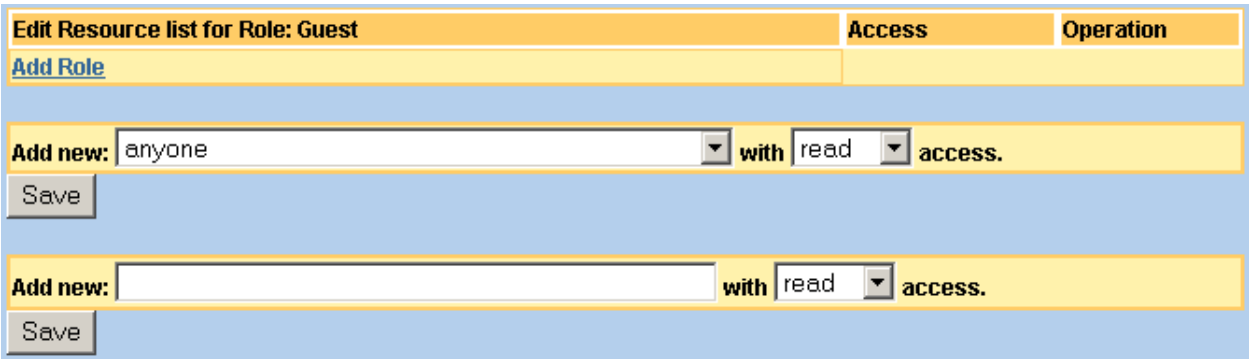
5. To exclude (disallow) the access to the configuration data for guest user, add with **exclude** option the resource *view.xml**.
6. Create *guest* user.
7. Assign role *Guest* for user *guest*.



8. To test and authenticate a user, you must add the created user ID with password to the web server administration, for example Tomcat Administration (*http://YourGateway:8080/admin*).

It is possible to assign multiple roles to a user. A role can have multiple resources. To identify the correct resource, you should make the request and look in the **System Log** with `debug`. The requested resource will be printed out.

Remember this resource string and assign it to the role. You can use the last dialog to copy and paste the resource name directly.



Caution: During the development of an access control list, you should not change your own role. If you delete your own rights to change the access control list, you will no longer be able to work. The delete or modify steps become effective immediately. Refer to **History of Configuration** for information on how to undo them.

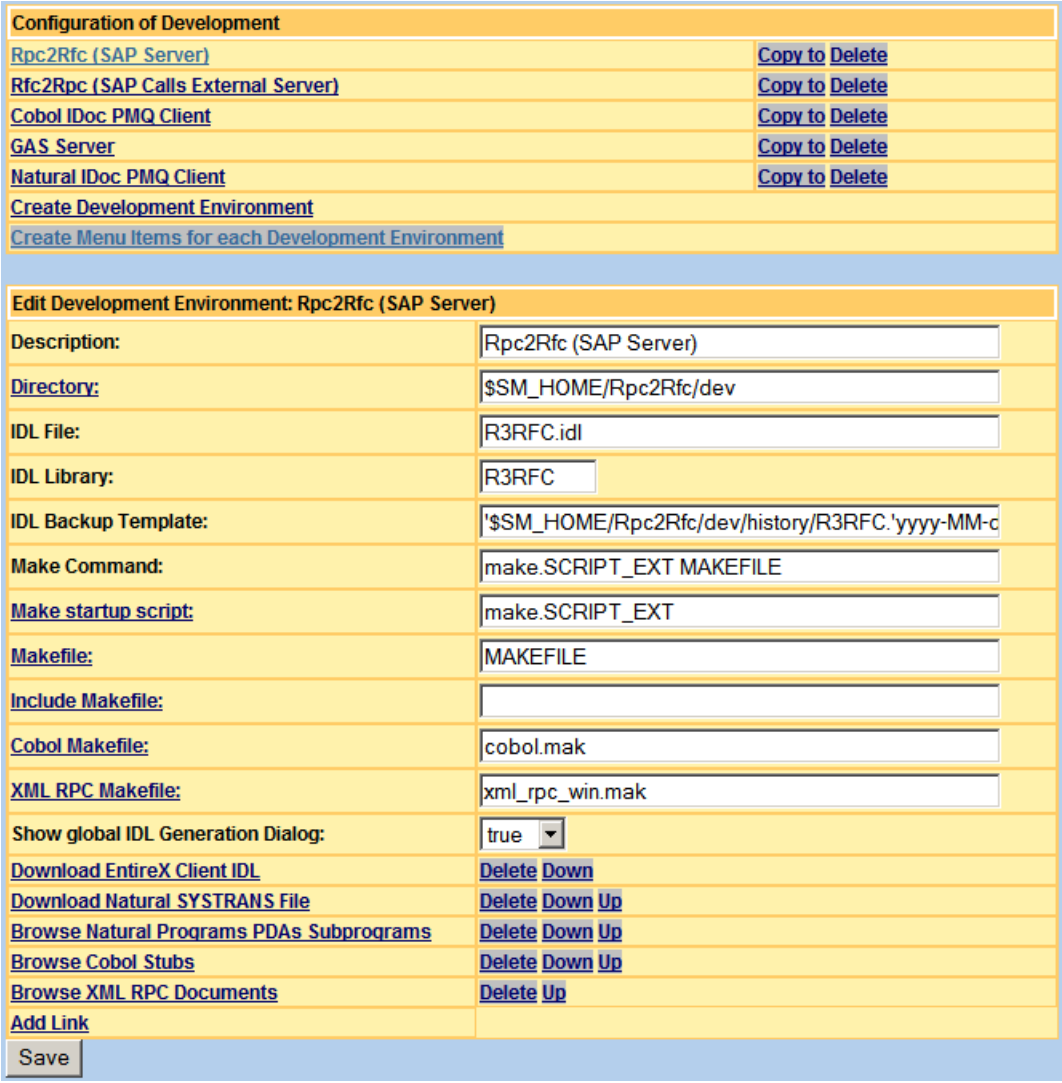
Tip: If you work with *anyone* resource, you can exclude explicit resources with the **exclude** option.

Tip: The wildcard character * is supported if you do not want to assign each resource. For example: The resource string `view.a*` allows access to all pages starting with character `a`.

Tip: Create your own administrator user ID with all rights as a fallback, before you make any changes to your own rights.

Development

The **development page** can be configured by choosing **Development** from the **Configuration** menu (*http://YourGateway:8080/sapr3gateway/manager/devAdmin*).



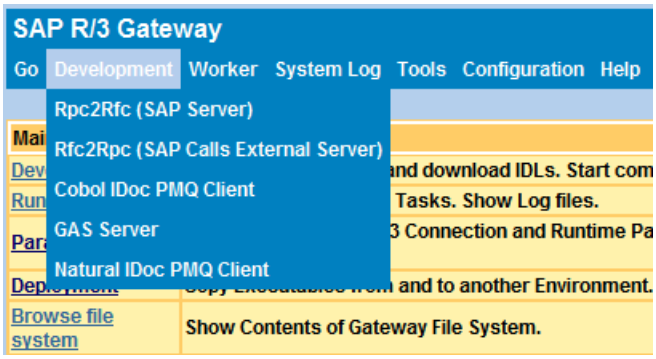
Configuration of Development	
Rpc2Rfc (SAP Server)	Copy to Delete
Rfc2Rpc (SAP Calls External Server)	Copy to Delete
Cobol IDoc PMQ Client	Copy to Delete
GAS Server	Copy to Delete
Natural IDoc PMQ Client	Copy to Delete
Create Development Environment	
Create Menu Items for each Development Environment	

Edit Development Environment: Rpc2Rfc (SAP Server)	
Description:	Rpc2Rfc (SAP Server)
Directory:	$SM_HOME/Rpc2Rfc/dev
IDL File:	R3RFC.idl
IDL Library:	R3RFC
IDL Backup Template:	'$SM_HOME/Rpc2Rfc/dev/history/R3RFC.'yyyy-MM-c
Make Command:	make.SCRIPT_EXT MAKEFILE
Make startup script:	make.SCRIPT_EXT
Makefile:	MAKEFILE
Include Makefile:	
Cobol Makefile:	cobol.mak
XML RPC Makefile:	xml_rpc_win.mak
Show global IDL Generation Dialog:	true
Download EntireX Client IDL	Delete Down
Download Natural SYSTRANS File	Delete Down Up
Browse Natural Programs PDAs Subprograms	Delete Down Up
Browse Cobol Stubs	Delete Down Up
Browse XML RPC Documents	Delete Up
Add Link	

Save

Each development environment allows you to define the directory where the IDL file and the makefile reside. We recommend creating a new development environment using the **Environment Wizard**, if you want to organize one IDL file for each project or organization department.

For fast access to development environments, the `Create Menu Items for each Development Environment` command extends the toolbar.



Development shows all environments. The submenu items only show the selected environment. To modify the toolbar, call **Menu Administration** (*http://YourGateway:8080/sapr3gateway/manager/menuAdmin*).

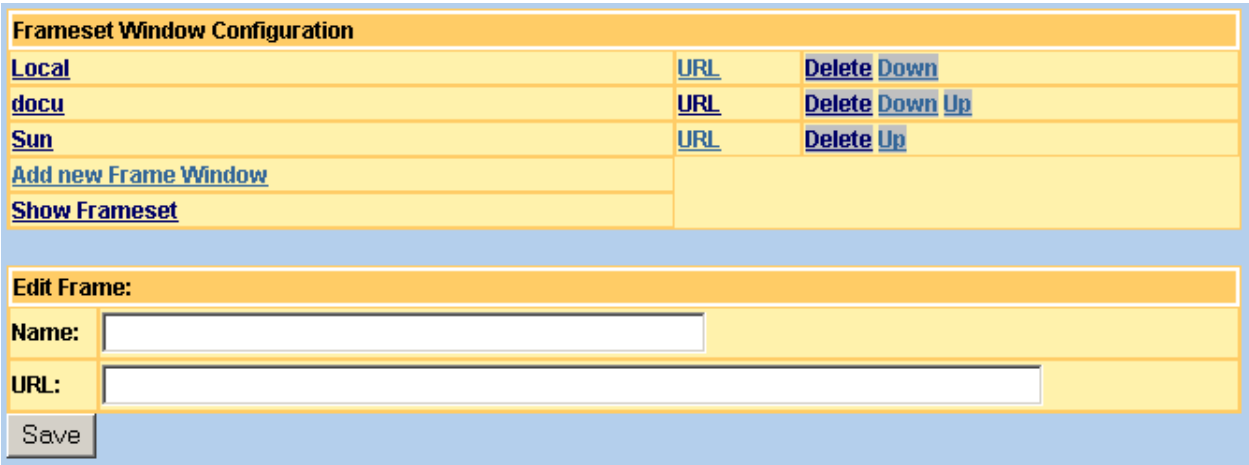
During the creation of a development environment you will be asked for a development type. The kind of parameters which can be defined depends on the development type. After the creation of a development environment you can define the following parameters

Parameter	Development Type	Description
Description	all	Title of development environment.
Directory	all	Directory in file system.
IDL file	all	Filename if the IDL file.
IDL Library	all	The library name which is defined in the **IDL library statement** in the IDL file.
IDL Backup Template	all	Template to create the history file on saving an IDL file.
Make Command	all	Executable command in directory to perfom the compilation process.
Make startup script	all	Filename of the executable command.
Makefile	all	Filename of main makefile.
Include Makefile	all	Filename of the included makefile of the main makefile.
COBOL Makefile	all	Filename of makefiles which are only statements for COBOL source code generating.
XML RPC Makefile	Rpc2Rfc	Filename of makefiles which are only statements for `.XMM` mapping generating.

Parameter	Development Type	Description
Show global IDL Generation Dialog	Rpc2Rfc	Value `false` suppresses the old IDL generation dialog. It prefers to use the new **IDL Editor**.
IDL Asynchronous File	Rfc2Rpc	IDL filename.
IDL Asynchronous Library	Rfc2Rpc	The library name which is defined in the **IDL library statement** in the IDL file.
IDL Asynchronous Backup Template	Rfc2Rpc	Template to create the history file on saving an IDL file.
Make Result (Running Task Command):	Rfc2Rpc	The result executable filename.
Add link	all	Add new link with URL and title to the development page for download or navigation about generated source code.

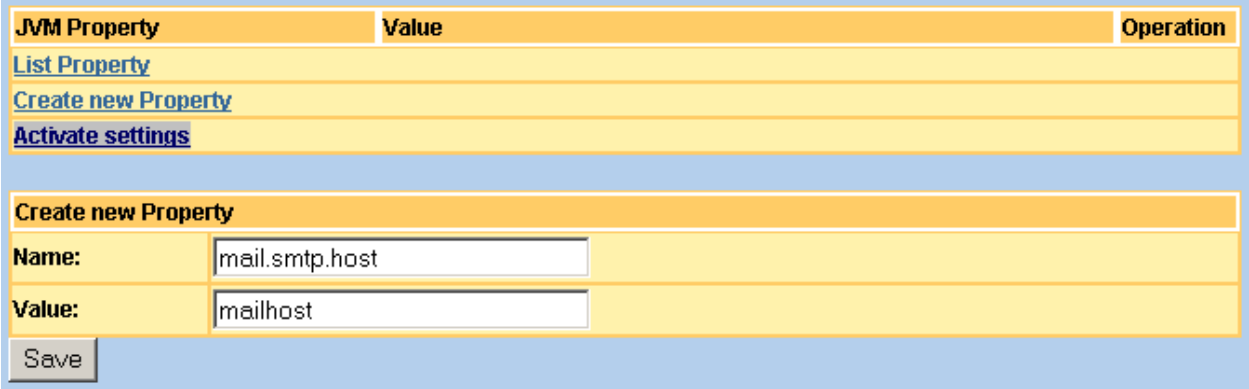
Frameset

A frameset can help you to handle multiple installations of the SAP R/3 Gateway web application in one browser window. There is an example in the section **Frameset** under the heading Optimization Tools. This menu allows you to define the number, sequence and URLs of a frameset.



JVM Properties

It is possible to set the Java Virtual Machine properties with the **JVM Properties** dialog from the **Configuration** menu. One main task of this feature is to set the IP-address (or DNS name) of an SMTP server for sending mails. The following example sets the `mail.smtp.host` to `mailhost`.

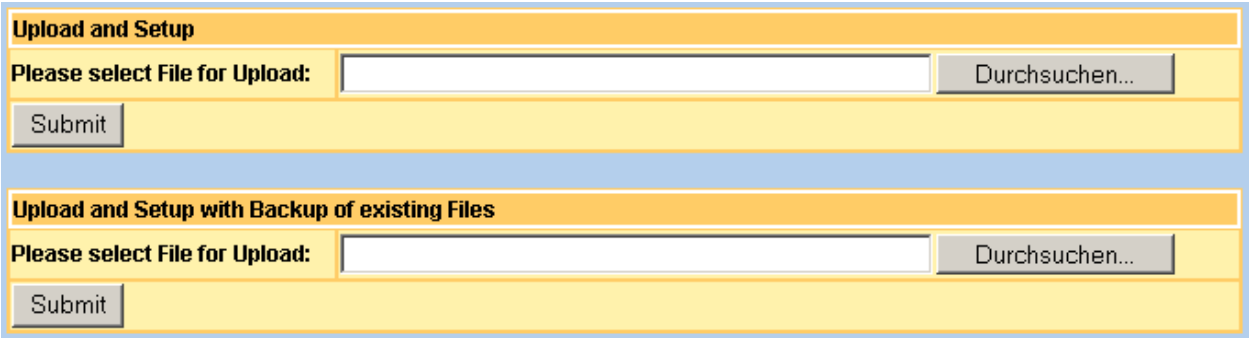


The properties are saved in the configuration file. When the System Manager is restarted, all properties will also be set. Before you create a new property, you should check whether it already exists. The **List Property** link lists all active JVM properties.

After making changes, you must activate them with **Activate settings**.

Upload and Setup

This feature allows you to transport one single file or one ZIP file from your desktop to the SAP R/3 Gateway web application. The **Upload** page on the **Configuration** menu (*http://YourGateway:8080/sapr3gateway/setup/upload*) allows you to select a file on your desktop.



The upload handles two kinds of file depending on the file extension:

- Files with ZIP extension are unzipped and all files contained in them are saved as in *webapps/sapr3gateway* directory.
- Files without ZIP extension are saved in the *webapps/sapr3gateway* directory.

You can select between two uploads:

- Upload all files without backupFiles .
- Backup all existing files before overwriting.

When the command has been finished, the page displays the result of the upload. If you select the Backup option, the created backup ZIP file name is displayed. This created ZIP file can be used for undoing an upload. The version information in *META-INF/MANIFEST.MF* is saved too. This means you have the version information in the created ZIP file and on undoing, the version information is restored.

To support multiple environments created by **Clone Environment Wizard**, the upload wizard asks the System Manager for configuration information. The URL of System Manager can be set to *http://YourGateway:8080/sapr3gateway/setup/SystemConstancy*. Set the value of `System Manager URL (SM_URL)` to `http://localhost:8080/sapr3gateway/manager`.

Clone Environment Wizard

This wizard provides support for step-by-step duplication of an existing environment.

If you wish to develop the IDL of **Rpc2Rfc** or **Rfc2Rpc** kernels in several departments or for different projects, we recommend cloning the current development, deployment and Running Task environments to a new one. It will then be possible to work with two or more IDLs and to have these executables running in parallel. The main advantage to this configuration is that the IDLs can be transferred to the production environment separately and independently. To provide support for multiple IDLs using this wizard, follow these main steps.

1. Copy the file system root directory of the selected kernel to a new one.
2. Add development, running task and deployment configurations to the System Manager.
3. Add a role for the access control list.

To install this wizard as a menu item, call the following URI *http://YourGateway:8080/sapr3gateway/manager/cloneEnvironment*. The first request creates the menu item, after the second request you will see the menu item **Clone Environment** in the **Tools** menu.

V

19 IDoc XML Gateway

The SAP R/3 Gateway provides the possibility to exchange documents asynchronously. SAP R/3 offers several document types, which are called IDocs. SAP developers can of course also define their own IDoc types. The following section helps you to exchange documents asynchronously between your application and SAP R/3 and vice versa.

This documentation defines the direction for exchanging IDocs for easier reading and understanding:

Outbound IDoc	An IDoc is created in your SAP R/3 application server and sent to IDoc XML Gateway. The IDoc XML Gateway will receive it.
Inbound IDoc	The IDoc XML Gateway sends an IDoc to your SAP R/3 application server.

The documentation of IDoc XML Gateway is organized under the following topics:

Overview and Design	Technical product information.
Distributation Model	Explains the SAP R/3 Distribution Model in a quick overview.
Network Considerations	Details about technical network communication.
Installation	Start with the installation and setup wizard.
Administration and Monitoring	The graphical view component for IDoc XML Gateway.
Configuration Parameter	Explains all settings for configuration parameters.
IDoc Type Information	Generates IDoc type information for transformation pipeline.
Using Pipelines	Configure and implement pipelines.
Natural, **COBOL**, **XSLT Extensions** and Java PMQ Stream and Server API	Framework for developing pipeline steps.

20 Overview and Design

The IDoc Gateway is designed for asynchronous document exchange. From a technical view, SAG's RPC and SAP's RFC protocol stacks are reused. Therefore, the IDLs of **Rpc2Rfc** and **Rfc2Rpc** have several additional functions.

Note: The IDLs of version 2.2.1 already contain these functions.

Function	Used in
IDOC_INBOUND_ASYNCHRONOUS	**Rpc2Rfc** uses this function to create an inbound IDoc in SAP R/3
IDOC_INBOUND_ASYNCHRONOUS	The **Rfc2Rpc** receives outbound IDocs delivered from SAP R/3.
IDOCTYPE_READ_COMPLETE	The function **Rpc2Rfc** returns type information about the structure of an IDoc

The IDoc XML Gateway provides functionality for sending and receiving IDocs asynchronously and for making the type information available to your business application. Your business application receives an interface for IDocs, which is generated from the type information.

The IDoc XML Gateway uses XML to represent the IDoc. Therefore, several XSL(T) transformations are available. The IDoc XML Gateway concatenates the transformation steps to a **pipeline** concept.

For the technical transformation of IDocs the **webMethods EntireX Broker's Concept of Persistent Messaging** is used. The basic concept is

- use of Units Of Work (UOW) for each IDoc and
- guarantee delivery in a transactional way.

The **pipeline concept** offers business views and hides the underlaying technical system.

21 Configuring an SAP R/3 Distribution Model

To handle inbound and outbound IDocs, you must define the Distribution Model in your SAP R/3. The following section helps you do this and gives an overview of how to use the transaction in SAPGUI. It will not explain every detail but it does provide a linked list to the necessary SAP transactions. This little workshop will help you to create an outbound "material logistics" IDoc.

Reuse of RFC Destination

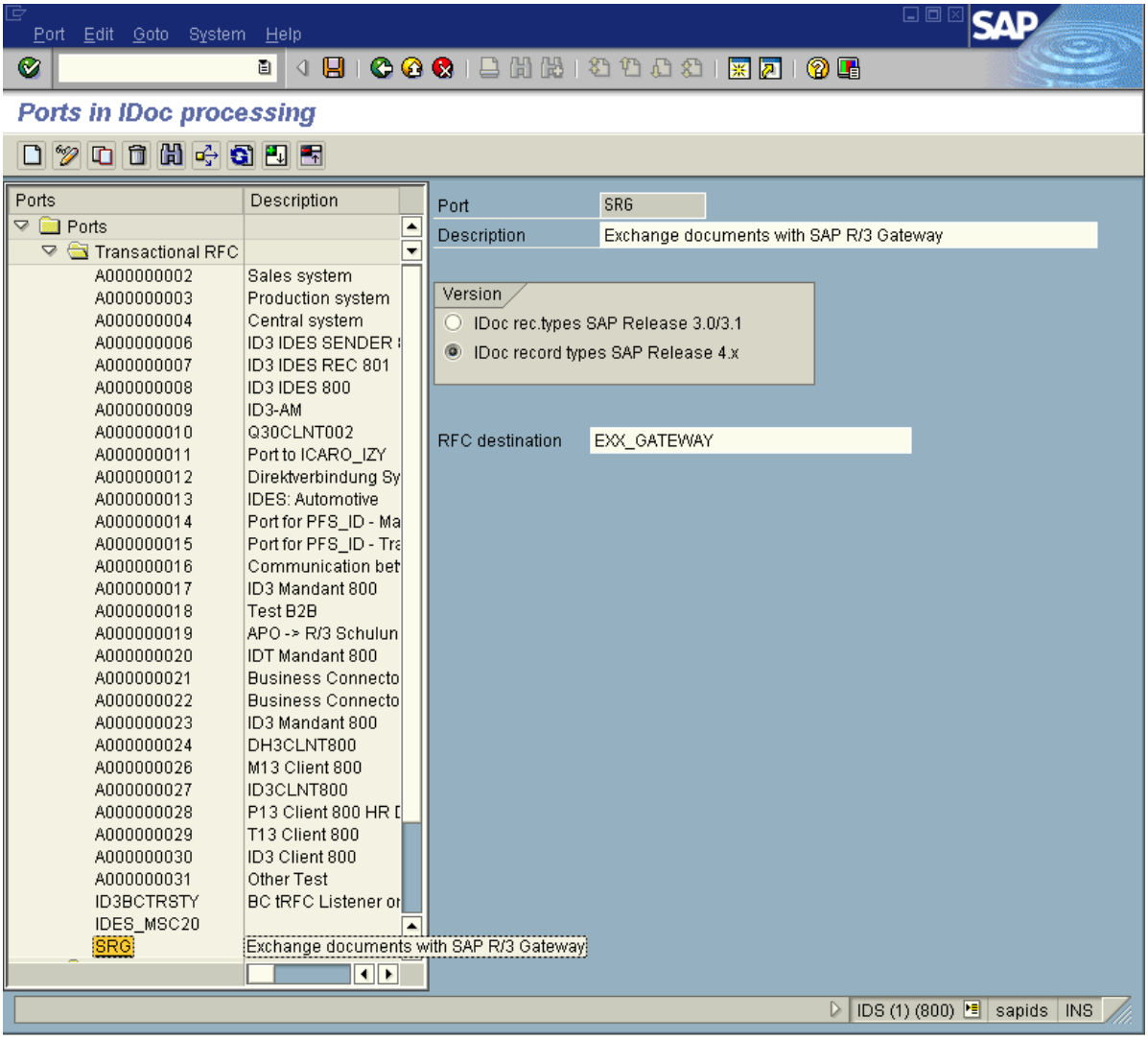
The RFC Destination which is defined for **Rfc2Rpc kernel** is needed to receive outbound IDocs. Remember the definition using SM59. Refer to the section **Running Task Rfc2Rpc (SAP calls External) Configuration** for more information.

Create Port for IDoc processing

The transaction `WE21` creates a port for IDoc processing. Select the port type `Transactional RFC`.

To create a port for IDoc processing

1 Start a transaction `WE21`.



2 Assign the created RFC Destination to this IDoc port.

Create Logical System

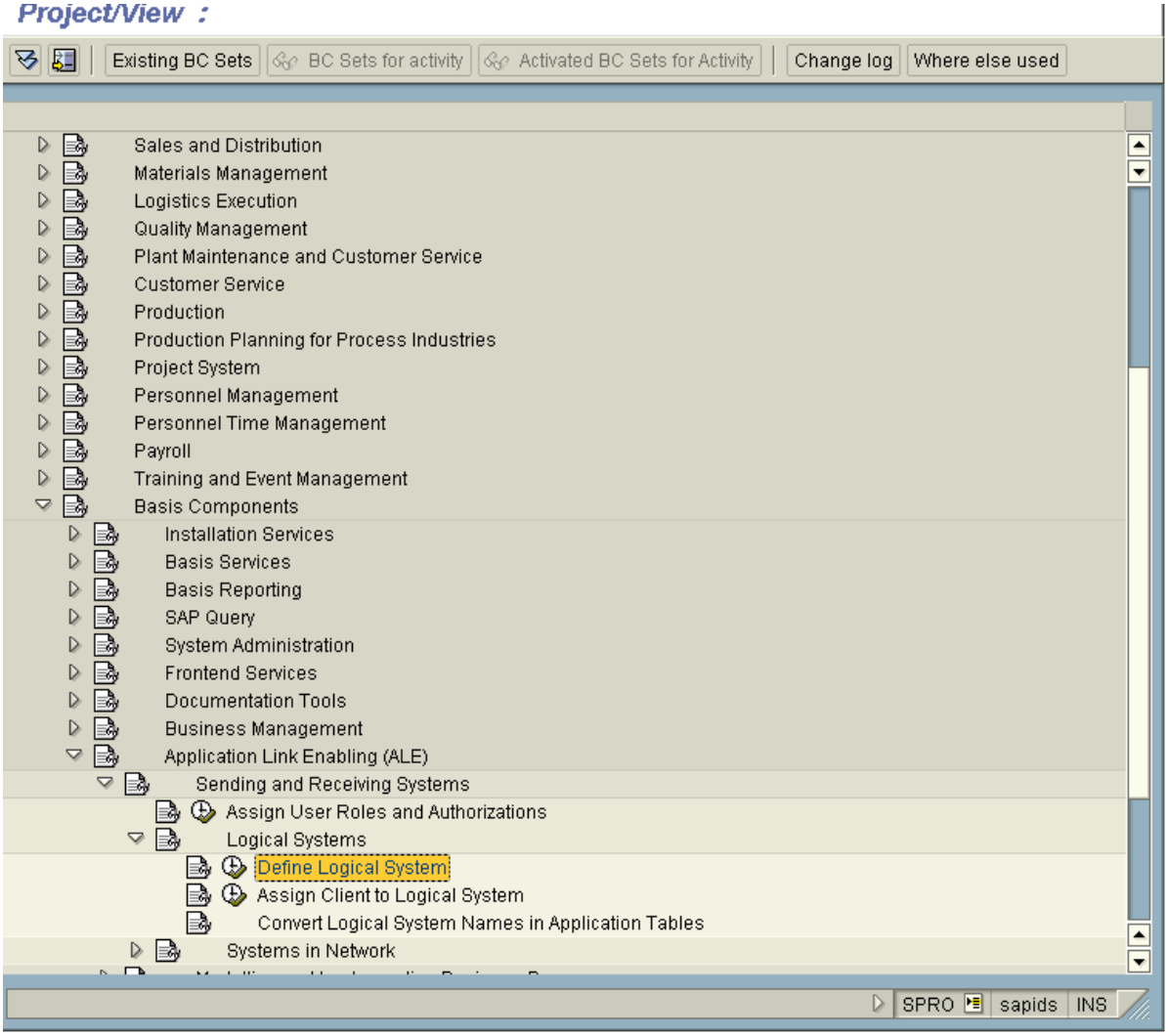
Use the transaction `spro` to create a logical system.

To create a logical system

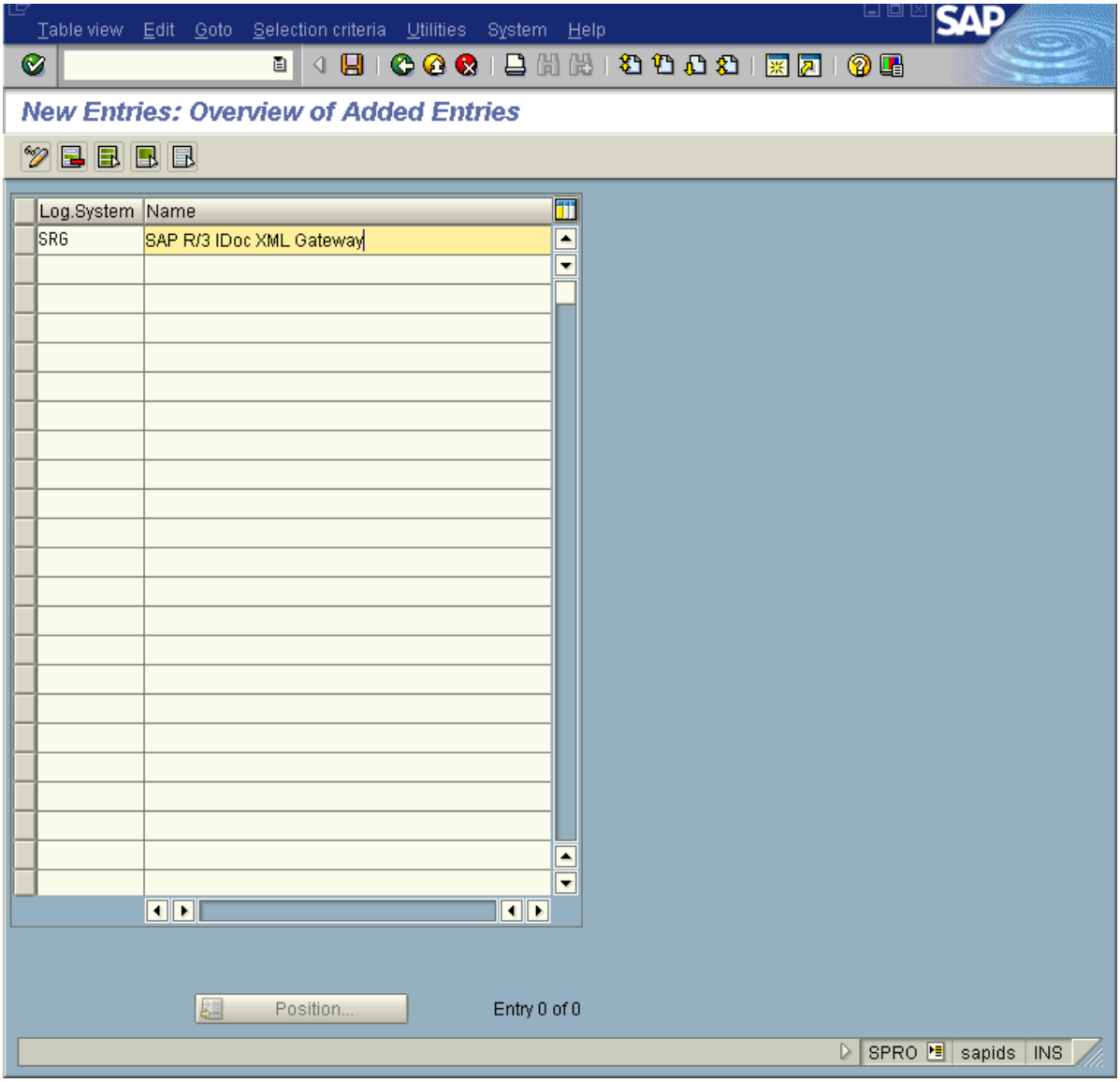
1 Start transaction `spro`.

2 Select **SAP Reference IMG**.

3 Expand the node **Define Logical System**. Choose **Basic Components**, **Application Link Enabling**, **Sending and Receiving Systems**, **Logical Systems** as in the screen snapshot below.



4 Using **New Entries**, create a new Logical System.



Create Outbound Partner Profile

The transaction `we20` creates a Partner Profile.

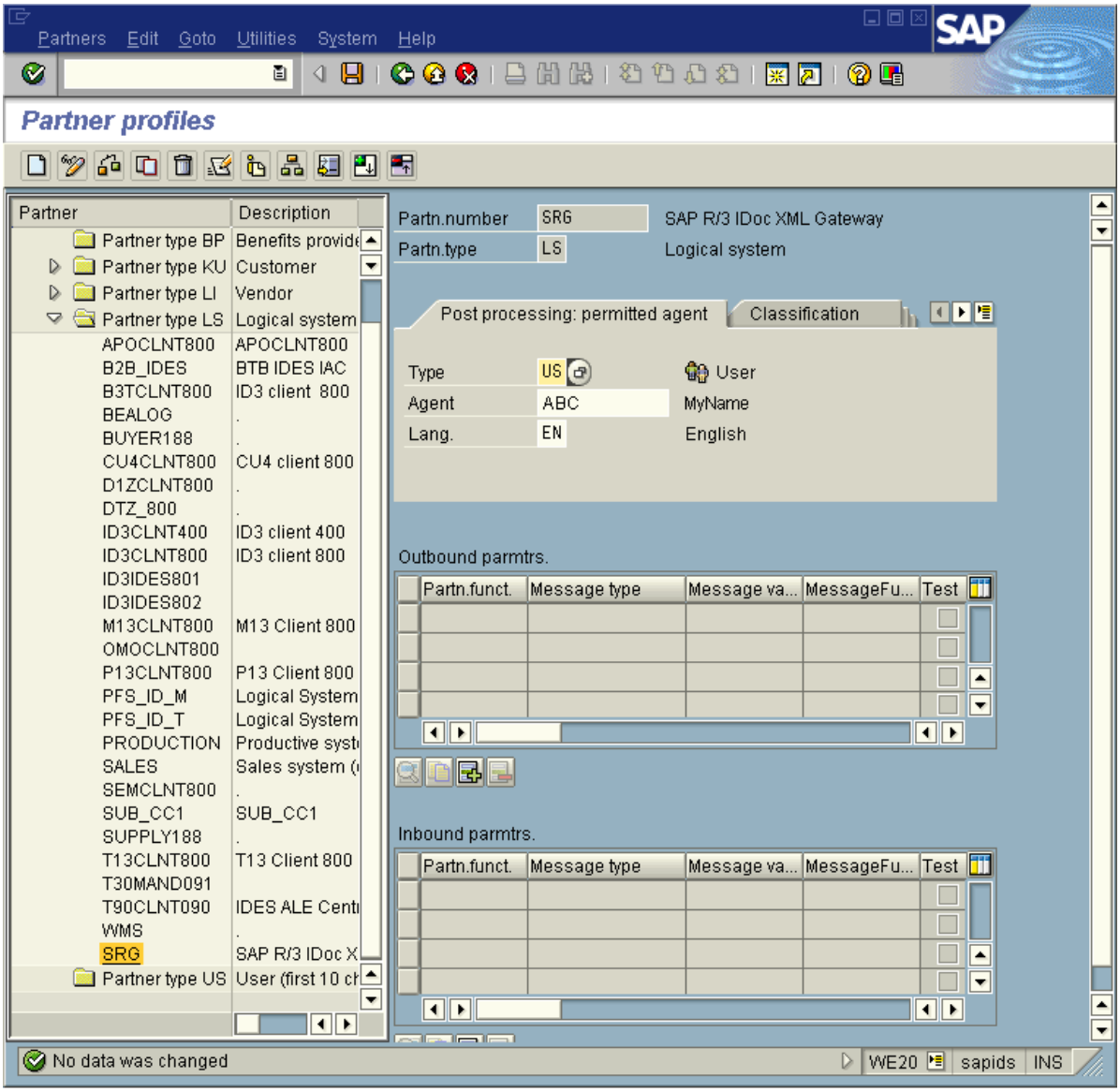
To create a partner profile with transaction `we20`

1 For **Partner Type** choose the value `Logical System` (`LS`).

2 For **Partner Number**, enter the value of your created Logical System. (e.g. `SRG`).

3 For **Type** select the value `User` (`US`).

4 For **Agent** enter your user ID for the SAP system.



The definition of the partner profile needs information about the IDoc types which are exchanged using this profile. The IDoc types must be defined for inbound and outbound.

To add an IDoc type "material logistics" for outbound processing

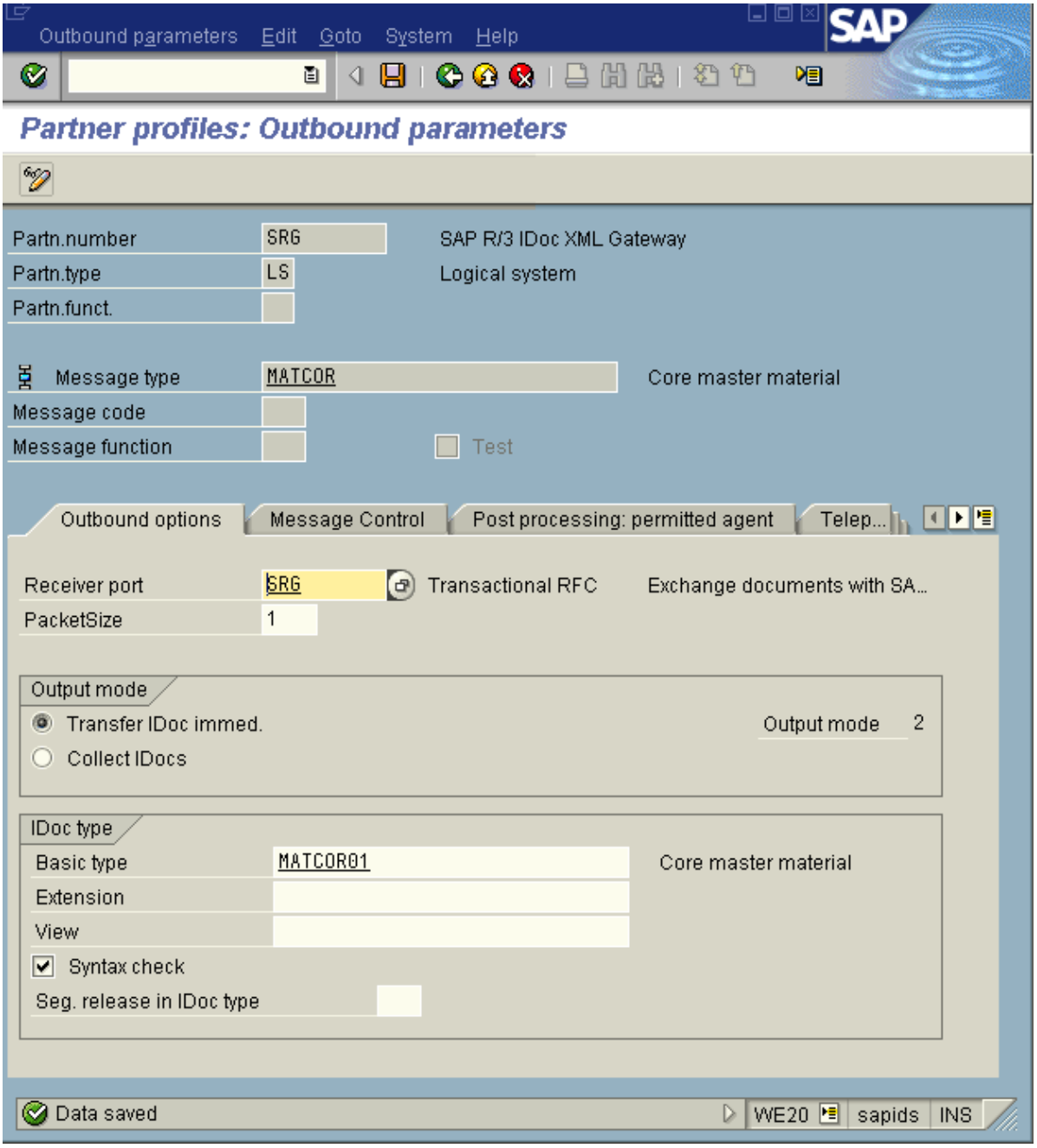
1 Set **Message type** to `MATCOR`.

2 Set the **Receiver port** to the IDoc Port you have created.

3 Use 1 for the **Packet size**. Currently only one IDoc per transaction is supported.

4 Set the **Output mode** to `Transfer IDoc immediately`.

5 Set the **Basic type** of generated IDoc to `MATCOR01`.



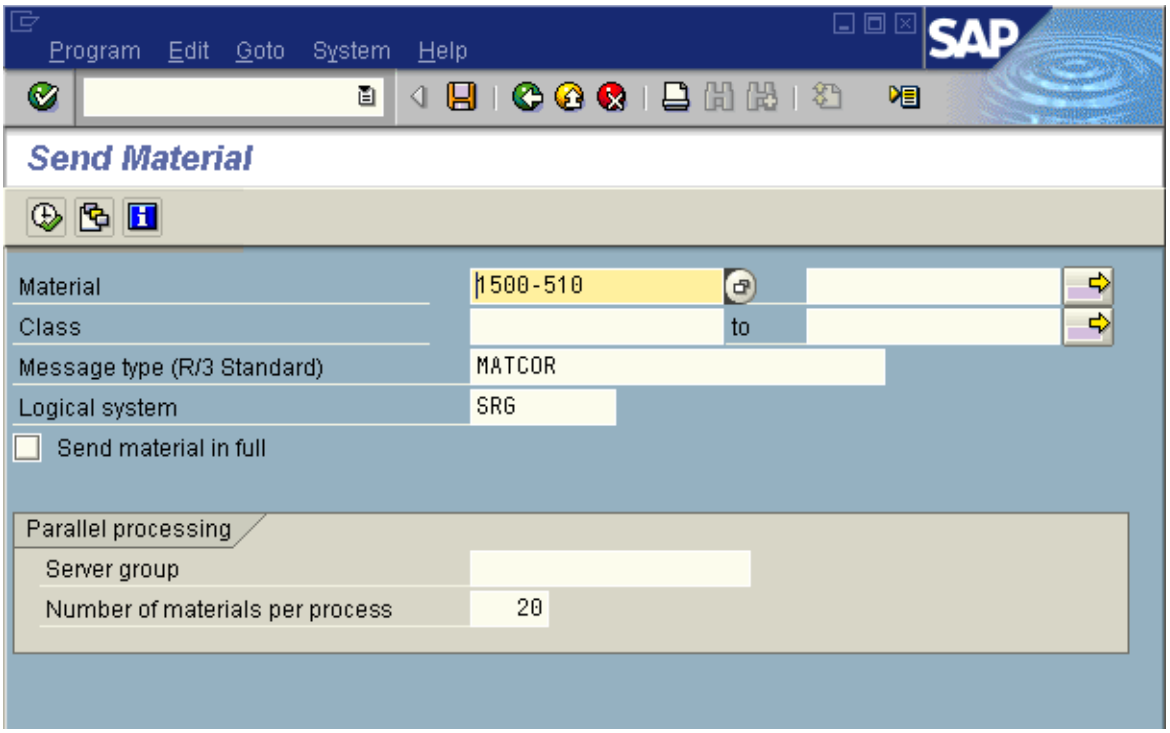
Test the Outbound Partner Profile

The transaction `bd10` creates an outbound material IDoc.

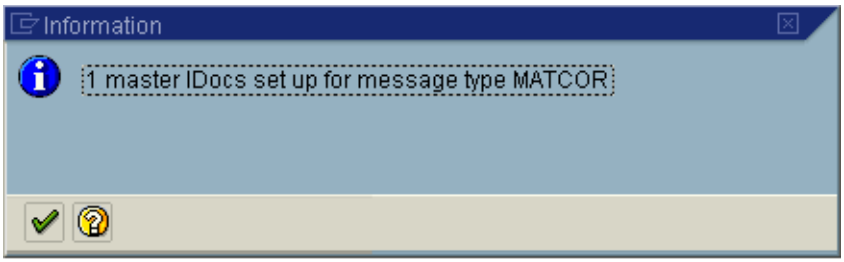
To test the outbound partner profile

1 Search for and choose a material with the ID.

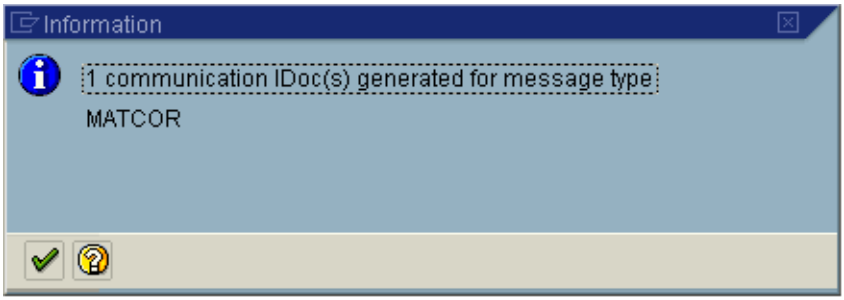
2 Enter the **Logical system** you created.



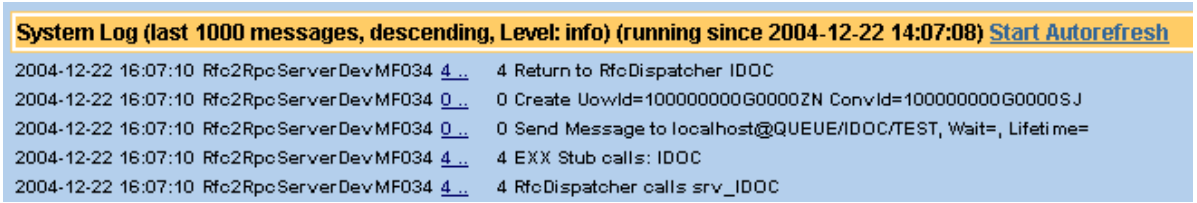
3 The system creates a material IDoc.



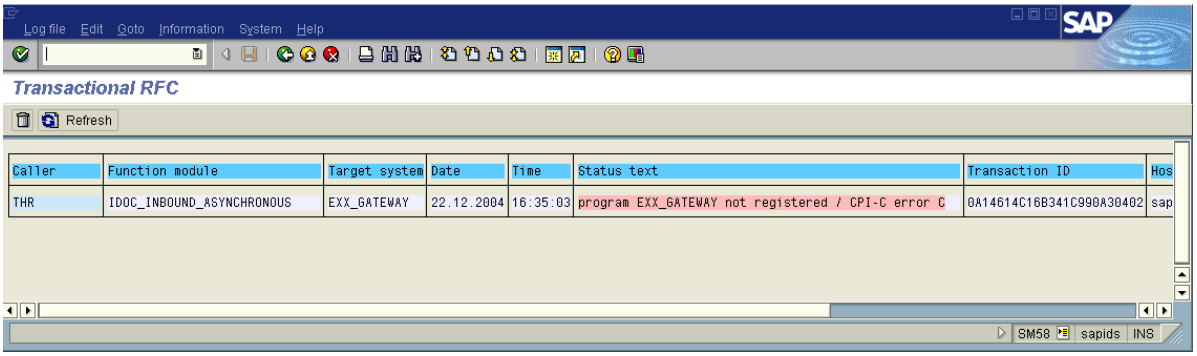
4 The system creates a communication IDoc.



5 A prerequisite for this step is the correct configuration of the Rfc2Rpc kernel (see the section **Configuring the Rfc2Rpc Kernel**). If the Rfc2Rpc kernel has been started, the IDoc will be received. See the section **Running Task Rfc2Rpc (SAP calls External) Kernel Environment** for details. The **System Log** shows the delivery process.



6 If the **Rfc2Rpc kernel** has not been started, restart the delivery with the transaction `sm58`. Select the line and press F6.



Change Pointers

For outbound IDocs, SAP R/3 provides a method to deliver changed information with mass data to external systems. For example, if logistic material mass data are changed, an IDoc with the changed information is generated and can be delivered immediately. Now it is possible to synchronize different databases. The following workshop explains how to enable the change pointers.

To enable pointer change

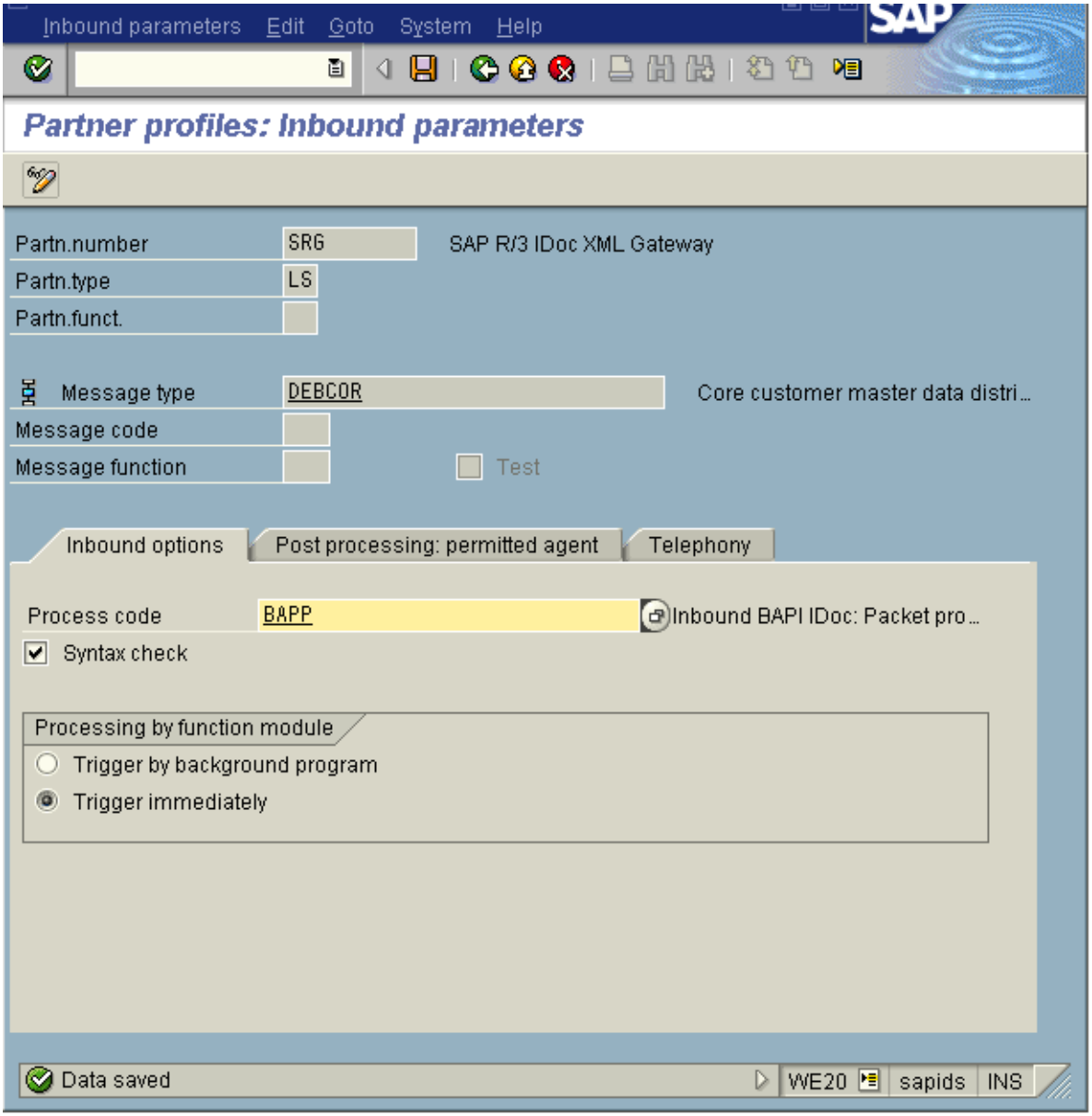
1. Call transaction `bd61` to perform a global activation.
2. With `bd50` activate your business message type.
3. Activate your specific field type with `bd52`. (This step is optional.)
4. Call a transaction to change a business object. For example: the transaction `mm02` allows you to change material data.
5. Check the creation of change pointers with transaction `se16` and table name `BDCP`.
6. Generate IDoc from change pointer running ABAP report `RBDMIDOC`.

Create Inbound Partner Profile

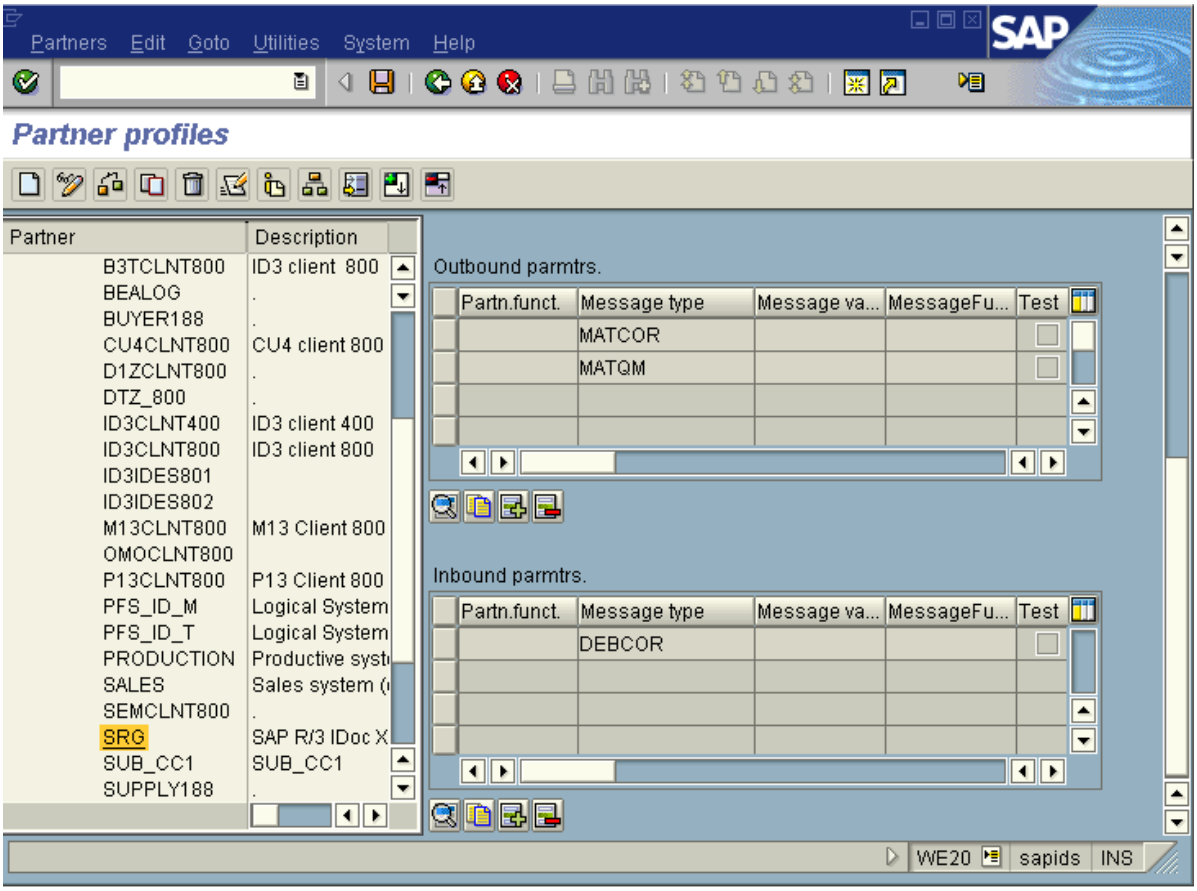
To receive IDocs from an external partner, the inbound partner profile must be configured. In addition to the **outbound partner profile**, the IDoc type must be set for an inbound partner.

To create an inbound partner profile

1. As with an outbound profile, start the transaction `we20` and select the created partner `SRG`.
2. Click the **Add** button to create an inbound parameter.
3. Set the message type. For example `DEBCOR`.



4 Save the changes and go back.



5 The SAP R/3 application server is now ready to receive IDocs of type `DEBCOR` with additional parameters for EDI_40 control record:

Parameter	Value
MESTYP	DEBCOR
SNDPRT	LS
SNDPRN	SRG

22 Network Considerations

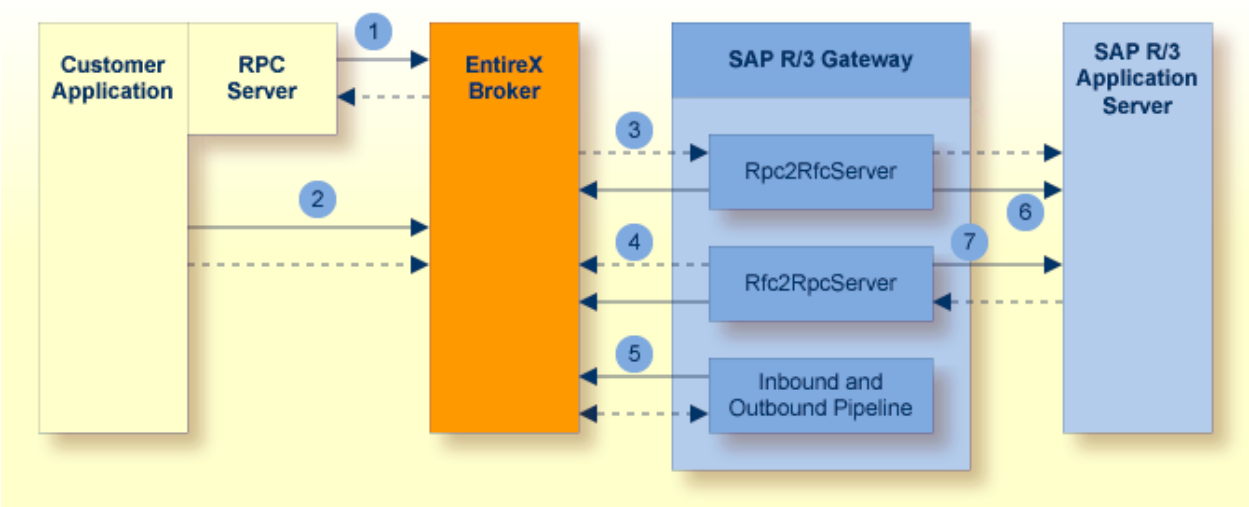
When communicating between applications which are running on distributed systems, you have to take the network into consideration. With the **pipeline concept**, the network consists of the following components:

- The message creator step for a pipeline.
- The message transfomer step.
- The message Broker.
- The message consumer step.

Each of the components communicates with its partner using TCP/IP. It is possible, for example, to optimize communication with the network using *localhost* channel. The following sections explain some network models. The network models differ depending on the location of the webMethods EntireX Broker. Ask the following questions for each network model

- Does a Broker already exist? If yes, it is possible to adapt the resources. See the section **webMethods EntireX Broker Parameters** for more details.
- What happens if one node is not reachable or temporarily not available?
- Do you have a WAN network connection between two nodes?
- Is encryption needed between two nodes?

The following picture shows the technical communication connection.



The communication partners are connected by solid and dashed lines in the image. The solid line defines the technical direction of communication. The dashed line shows the logical data flow.

1. The RPC Server for your custom application creates a network connection to the EntireX Broker and waits for requests. The network protocol TCP/IP can be used for network connection. On mainframe it is possible to use the SVC communication change to a local EntireX Broker on the same node.
2. Your custom application creates an RPC call with synchronous communication or an ACI call with asynchronous communication to send IDoc documents to the EntireX Broker. TCP/IP is used or SVC on mainframe for a local Broker.
3. The Rpc2Rfc kernel creates a TCP/IP connection to the EntireX Broker and waits for requests.
4. The Rfc2Rpc kernel forwards incoming requests to the EntireX Broker. The TCP/IP connection is created automatically.
5. The pipeline steps communicate and create TCP/IP sockets to the EntireX Broker with TCP/IP.
6. The Rpc2Rfc kernel forwards incoming requests to the SAP R/3 application server. The RFC protocol with TCP/IP is used.
7. The Rfc2Rpc kernel creates a TCP/IP connection to the SAP R/3 application server and waits for requests. An incoming request is forwarded to the EntireX Broker.

Notes:

1. The EntireX Broker is passive and does not create or initialize any communication.
2. Each communication creator must have the IP address or DNS name of his partner.

To optimize communication with the EntireX Broker consider these three models:

Model	Valuation
Broker on SAP R/3 Gateway Node	There is a network connection from your mainframe mission-critical application to an external SAP R/3 Gateway node. IDoc documents will be sent over the network.
Broker on Application Node	On mainframe it is possible to use local communication via SVC.
Broker on all Nodes	It is possible to use the best EntireX Broker depending on whether communication is asynchronous or synchronous.

23 Installation

The IDoc XML Gateway runs as a web application called *sapr3idocxmlgateway*. A prerequisite is the installation of SAP R/3 Gateway (see the section **Installing the Gateway Portal**). If you already have an older version, you must migrate it as described in the section **Migrate to 2.3.1**. After migration or a new installation, you can proceed to the following sections.

Web Application Server

To install IDoc XML Gateway

- Use Tomcat Manager to upload *sapr3idocxmlgateway.war* from the installation medium (CD-ROM). See **Installing the Gateway Portal** for information on how to complete this step with the main SAP R/3 Gateway application.

  Or:

  Copy *sapr3idocxmlgateway.war* to the Tomcat sub-directory *webapps*.

  Note: If you wish to use JBoss as web application server, you must install *sapr3idocxmlgateway.war* in the same way in which SAP R/3 Gateway is installed. Refer to the section **Using JBoss** for more details.

Setup Wizard

For post installation steps, start the setup wizard on *http://YourGateway:8080/sapr3gateway/manager/setupWizardIDocXMLGW*.

The setup wizard helps you to configure the SAP R/3 Gateway for the IDoc XML Gateway. In addition to the wizard, perform the configuration steps described in the next section, **Configuration Parameters**.

24 Configuration Parameters

The following parameter list describes the system requirements for IDoc XML Gateway.

Web Application Server

The transformation from IDoc to XML and vice versa needs memory in your web application server. The default memory settings of the JVM are insufficient.

If the web server runs with SAP R/3 Gateway and IDoc XML Gateway alone on the gateway machine, you can assign 70% of the real memory to JVM. For example, if the gateway machine has 1GB of real memory, assign 700MB to the JVM of the web application server

To set the avaliable memory for the JVM, use the -Xms (for initialization) and -Xmx (for maximum) options. Depending on your platform, set the parameters as described in the table below.

Location	Description
Tomcat on UNIX	Add or set the *JAVA_OPTS* environment variable for memory requirements in the startup script as described in the section **Startup at Boot Time**.
Tomcat on Windows defined as service	Add parameters for JVM in the registration database (regedit) on the path *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Apache Tomcat 4.1\Parameters*
Tomcat on Windows with manual startup	Add or set the *JAVA_OPTS* environment variable in *startup.bat* of Tomcat's bin directory.
JBoss on UNIX	Add or set the *JAVA_OPTS* environment variable in the script that calls *run.sh* in JBoss's bin directory.

Length of Received Messages

To receive large outbound IDocs from SAP R/3, the maximum message size must be set on the **JVM Property** page. Add a property with the name `pmq.resource.maxReceiveLength` and the value `100000`. This length must in any case be greater than the **internal buffer size of Rfc2Rpc kernel**.

In case of error, if you do not set (or to less) this parameter or set it to a value less than the internal buffer size of the Rfc2Rpc kernel, the message `API: Msg truncated to fit receive-buffer` will occur in the **System Log**.

Configuring the Rfc2Rpc Kernel

The **Rfc2Rpc kernel** receives the outbound IDoc and delivers it to an **webMethods EntireX Broker Message Queue**. The following parameters must set for message handling.

- Internal Buffer
- IDoc Queue
- IDL

Internal Buffer

The Rfc2Rpc kernel needs a buffer size large enough to generate a UOW (Unit of Work) message. This buffer is defined statically on the stack at compilation time. To define the buffer size add the following CFLAG compile parameter `-DPMQ_SEND_BUFFER_LENGTH=<Size in Bytes>` to the makefile.

```
CFLAGS= -c \
 -DPMQ_SEND_BUFFER_LENGTH=100000 \
...
```

The default is 32000 bytes if this CFLAG parameter is not set. If the generated message is larger than the defined buffer size, the UOW will contain multiple messages. In this case, check the webMethods EntireX Broker parameter UOW-MSGS and increase its size if necessary (refer to the section **webMethods EntireX Broker Parameters** for details).

IDoc Queue

To address the message queue

- In the Rfc2Rpc kernel, set the parameter described in the table below. Refer to the section**Running Task Rfc2Rpc (SAP calls External) Kernel Environment** for more details.

Parameter	Environment Variable	Description
webMethods EntireX PMQ Server address	EXX_PMQ_SERVER_ADDR	Send asynchronous units of work to this webMethods EntireX Broker address. Use the value `localhost@QUEUE/OUTBOUND-IDOC/TEST` to send the IDoc queue of localhost Broker.

This **parameter**, the **EntireX Client User ID** and the **EntireX Client Password** are reused for identification.

IDL

To work with one IDoc in one UOW (Unit of Work), the working memory size of the function IDOC_INBOUND_ASYNCHRONOUS must be increased. Edit the **asynchronous IDL of Rfc2Rpc kernel** on the development page.

- The index of array `IDOC_DATA_REC_40` corresponds to the segments of an IDoc. Set the index to the maximum number of segments, for example to 100.

```
1 IDOC_DATA_REC_40 (/1:100) In /* EDI_DD40 ↩
```

- The index of array `IDOC_CONTROL_REC_40` must be changed to 1, because only one IDoc can be received per transaction.

```
1 IDOC_CONTROL_REC_40 (/1:1) In /* EDI_DC40
```

webMethods EntireX Broker Parameters

Asynchronous communication needs some configuration parameters in the **webMethods EntireX Broker Attribute File**. To handle IDocs like units of work (UOWs) in persistent store (refer to the webMethods EntireX documentation **Concepts of Persistent Messaging (Primary Store)**), the following parameters must be checked in the existing Broker or set for a new Broker. For more information refer to the topic **Attributes used for Unit of Work** in the webMethods EntireX Documentation "Using Persistence and Units of Work".

Parameter	Recommended Value	Description
PSTORE	HOT	Start Broker with restore.
DEFERRED	YES	Server does not have to be active.
STORE	BROKER	Store UOWs persistent
UWTIME	3D	Define the lifetime for UOW.
UWSTATP	1	Remember UOW status.
MAX-UOWS	100000	Max UOWs
UOW-MSGS	100000	Max number of messages in one UOW.
MAX-MSG	1000000	Defines the maximum message size.
NUM-LONG-BUFFER	MAX-UOWS * SizeOfOneUOW / BlockSize	Defines the number of blocks to receive and to keep in the UOWs in memory.
NUM-SERVICE	Add 20 to existing	Defines the number of services (=queues). Calculate 7 per environment (test, integration, production).
NUM-SERVER	Add 20 to existing	Defines the number of servers. Calculate 7 per environment (test, integration, production).

All parameters are set as default or as maximum in the section **Broker-specific Attributes**. In addition, you can redefine the parameter settings for each service in the section **Service-specific Attributes**.

Note: webMethods EntireX has two modes for handling service-specific attributes. In **service update mode**, service-specific attributes can be changed during runtime; in non-update mode, the Broker needs to be restarted to effect any changes. Refer to the webMethods EntireX documentation **Broker Attribute File (All Platforms)** for further details.

Add the following lines to define the queues in the service-specific attributes (refer to the **Broker Attribute File (All Platforms)** for more detail):

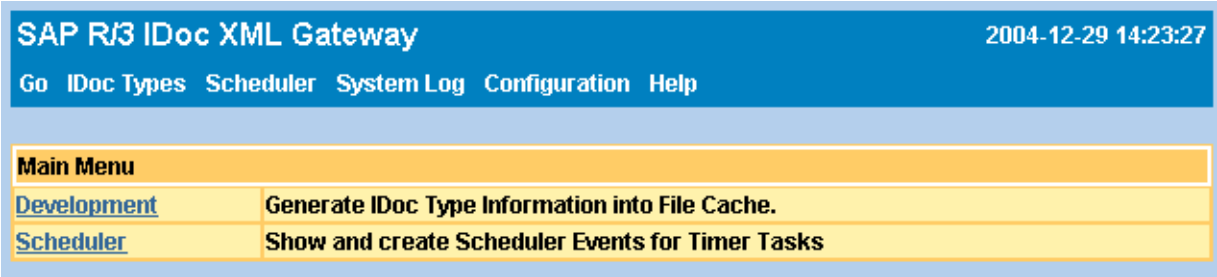
```
DEFAULTS         = SERVICE
   CONV-NONACT   = 5M
   SERVER-NONACT = 5M
   TRANSLATION   = SAGTCHA

   CLASS = QUEUE ,   SERVER = * ,   SERVICE = TEST   * Test environment
   CLASS = QUEUE ,   SERVER = * ,   SERVICE = INT    * Integration environment
   CLASS = QUEUE ,   SERVER = * ,   SERVICE = PROD   * Production environment
```

Tip: If you decide to use one Broker for all environments (test, integration and production) then define a resource limit for each queue environment. For example: The following line defines only a limit of 100 UOWs for each TEST queue. Add the additional parameter for restriction to the same line of service.

```
CLASS = QUEUE ,   SERVER = * ,   SERVICE = TEST , MAX-UOWS = 100
```

25 System Manager

The System Manager of IDoc XML Gateway looks the same as for the **SAP R/3 Gateway**



The following features are adapted for IDoc exchange:

- The **System Log**
- Only the Worker **Scheduler** is configured to handle the log files.
- The development contains the feature to generate the **IDoc type** information.
- The **Help** menu shows version information and goes to this documentation.
- To upload changes and fixes, the **Upload feature** is enabled.

The System Constants page has several parameters for logging:

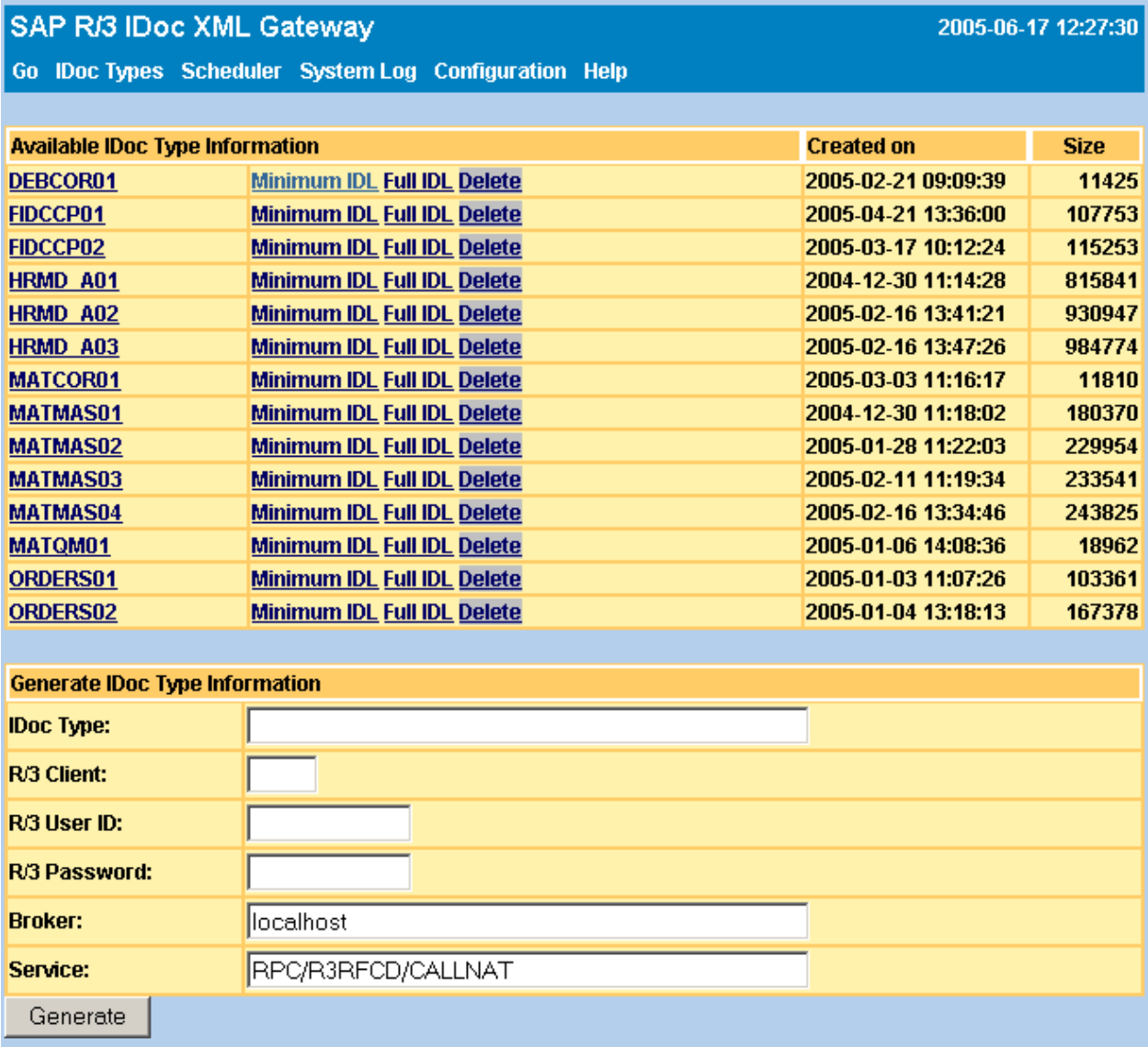
System Constancy		
Title:		SAP R/3 IDoc XML Gateway
Show Memory Usage:		false
System Manager Home Directory:	$SM_HOME	/FS/fs0225/home/thr/tomcat/webapps/sapr3idocxmlgateway
System Manager URL:	SM_URL	http://localhost:8080/sapr3idocxmlgateway/transforme
EntireX XML-RPC URL:	XML_RPC_URL	http://localhost:8080/sapr3idocxmlgateway/adapter
Mail to Gateway Administrator:	MAIL_TO_GW_ADMIN	
Log default Life Time in days:	LOG_DEFAULT_LIFE_TIME	5
System Manager Log Level:		info Activate
System Manager Log View of max. Message Length:		1000 in bytes. Activate
System Manager Log Home directory:		$SM_HOME/log
System Manager Log Filename Prefix:		sapr3idocxmlgateway
System Manager Log File Life Time:		10 in days. Activate
System Manager Log Limit of cached Messages:		100
System Manager Log Display Sort Order:		descending
System Manager Log auto refresh Time:		5 in seconds.
System Manager Log flush immediately:		true Activate
Server Logger URL:		http://localhost:8080/sapr3gateway/manager Activate
Server Logger User ID:		
Server Logger Password:		
Server Logger Resource:		IDocGW.

Save

In addition to the IDoc XML Gateway System Log, the messages are sent as a copy to the SAP R/3 Gateway web application. The parameter **Server Logger URL** defines the endpoint URL. As a result, there is one location for all messages.

26 Generating IDoc Type Information

Some transformation processes need IDoc type information. For example, the outbound IDoc is received as plain text. The subsequent transformation process must interpret the plain text. The generation process calls the Rpc2Rfc function `IDOCTYPE_READ_COMPLETE`. The transformation process in the **pipelines** needs the IDoc type information. To generate the specific IDoc type information, go to the development **IDoc Type** page of IDoc XML Gateway or to *http://YourGateway:8080/sapr3idocxmlgateway/transformer/IDocType_ToFile*



The first table shows the **Available IDoc Types** that have already been successfully generated by SAP R/3. The **Delete** button will delete the corresponding type. The **IDL** button generates the webMethods EntireX interface definition in a new window.

To start the generation process, you must set the fields in the **Generate IDoc Type Information** dialog:

Field	Default	Description
IDoc Type	None	Enter the IDoc type name.
R/3 Client	None. See **Rpc2Rfc** parameter **Gateway Logon Strategy**	Connect with R/3 client.
R/3 User ID	None. See **Rpc2Rfc** parameter **Gateway Logon Strategy**	Connect with R/3 user ID.
R/3 Password	None. See **Rpc2Rfc** parameter **Gateway Logon Strategy**	Connect with R/3 password.
Broker	localhost with default port 1971	IP connection parameter
Service	RPC/R3RFCD/CALLNAT	Name of broker service

The generation process waits until all data have been retrieved from the SAP application server. The next page will show an update of available IDoc types or the **System Log**, if an error occurs. In this case, the first two lines will show the error message.

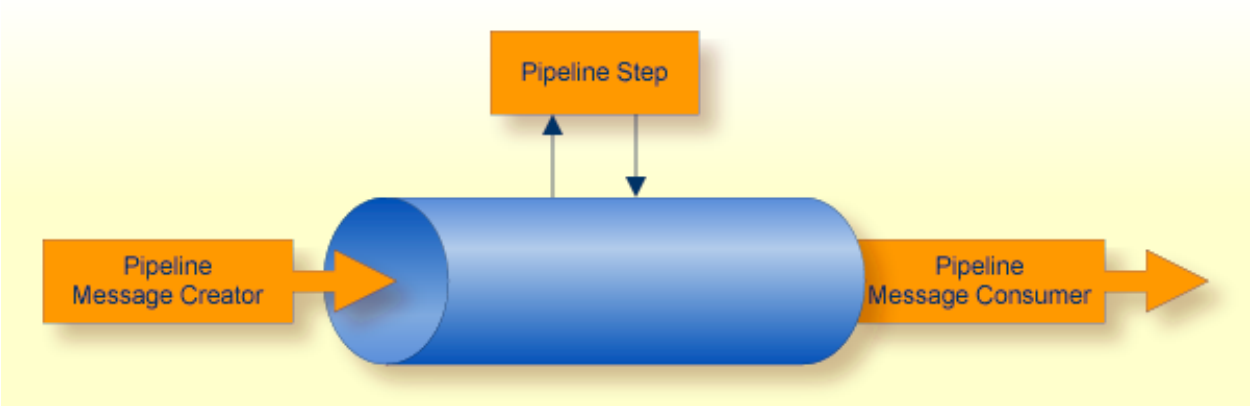
Tip: Type information contains many comments on the groups and fields. The comments are language dependent. You can change the language parameter on the **Rpc2Rfc kernel** to get the comments for your language.

Important: **Pipeline** processing stops if an IDoc type is missing. If this happens, generate the IDoc type and restart the pipeline step for transformation for processing to continue.

The IDoc types in the file system work for the transformation step as a cache. It is possible to change the default handling and to get the IDoc type for each transformation step at runtime from the SAP R/3 system. Note however, that this requires considerable resources and will have an effect on performance. To change the default file cache handling, edit the *IDocType.xsl* stylesheet. Change the documented `xsl:import` statement.

27 Configuring and Implementing Pipelines

A pipeline transports the IDoc messages from a creator to a consumer. During transport, there may be several steps for transformation. A pipeline step works transactionally. The step must be committed or backed out. When a step is backed out, it will be performed later. When a step is committed, the pipeline continues processing and the next pipeline step is called.



From a technical or Broker view, a pipeline contains many queues. The pipeline page *http://Your-Gateway:8080/sapr3gateway/manager/pipelines* gives an overview of all pipelines.

Pipelines	Status	Operation
IDoc Inbound	Show	Delete Down
IDoc Outbound	Show	Delete Down Up
Logistic Data Flow	Show	Delete Down Up
IDoc XML	Show	Delete Up
Create new Pipeline		

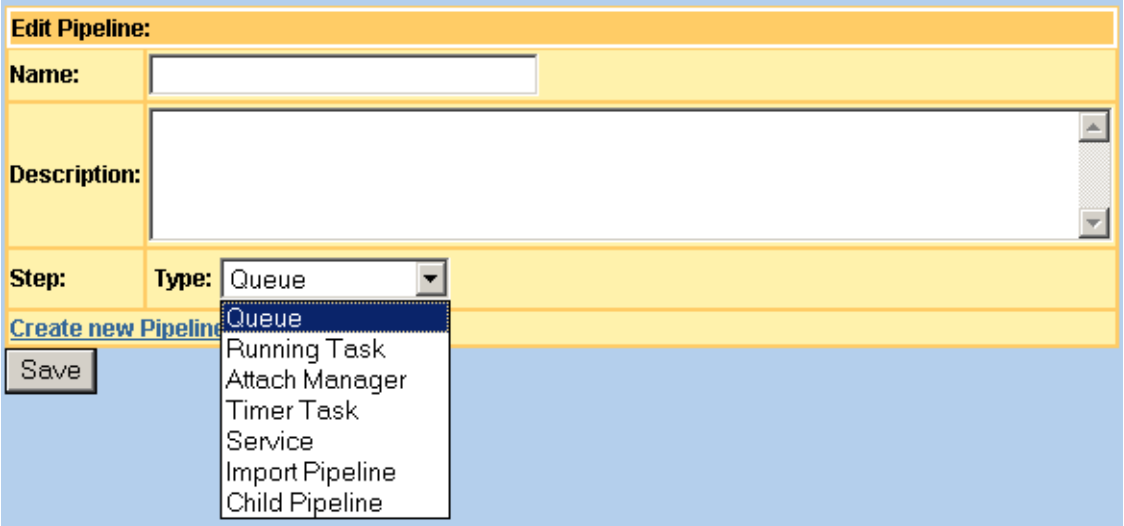
You can scroll down

- to create and delete pipelines,
- to create and change pipeline steps (press link for selected pipeline)
- to see the status and contents of a whole pipeline and a pipeline step. Use the **Show** link.

Creating a Pipeline

It is possible to create your own pipeline on *http://YourGateway:8080/sapr3gateway/manager/pipelines*.

To create a new pipeline view, press the **Create new Pipeline** link.



The next dialog asks for more parameters depending on the selected pipeline step.

Parameter	Description
Name	Name of the pipeline.
Description	Document the pipeline or a pipeline step.
Create new Pipeline Step	Opens the dialog to create a new step at the end of the pipeline. You will be prompted for the type of step. After selecting the type, press the **Save** button. You will be prompted for more parameters depending on the type selected.
Delete	Delete this pipeline step
Up	Move the position of this step up
Down	Move the position of this step down
Type: Queue	Shows the status information of Broker queue. The broker address and queue name are needed as additional parameters to retrieve the information. For example: the number of accepted UOWs is displayed. Depending on **EntireX Broker UWSTATP parameter**, the number of processed, backouted or canceled UOWs are displayed. **Tip:** This pipeline step takes a long time on many (greater than 1000) UOWs. Use **Queue Info Service** pipeline step for a quicker response time.
Type: Queue Info Service	Shows the number of accepted UOWs, the lifetime and the limit number of UOWs for this service.
Type: Running Task	Shows the status information of the selected **Running Task**. The dialog shows the available tasks that can be selected. **Start** and **Stop** operations are still available.
Type Attach Manager	Shows the status information of the selected **Attach Manager**. The dialog shows the available items that can be selected. **Start**, **Stop** and **Simulate** operations are still available. **Note:** The **Simulate** process can only run if the **Attach Manager** is or was running.

Parameter	Description
Type: Timer Task	Shows the status information of the selected **Timer Task**. The dialog shows the available items that can be selected. **Start**, **Stop** and **Simulate** operations are still available.
Type: Service	Shows the EntireX Broker service information. The broker address and service name (CLASS/SERVICE/SERVER) are needed as additional parameters to retrieve the information. For example: The number of active servers is displayed.
Type: Import Pipeline	Shows the referenced pipeline in the current pipeline. The dialog shows all of the pipelines that are available. **Caution:** The current pipeline is disabled, but, you should not build a cycle.
Type: Child Pipeline	Shows the referenced pipeline as a child of this pipeline. The dialog shows all of the pipelines that are available. **Caution:** The current pipeline is disabled, but you should not build a cycle. **Tip:** This feature allows you to build a business pipeline for your complete application. A subsequent child pipeline is in this case an outbound and an inbound pipeline.
Queue Message Contents	Shows the command for reading the next UOW message and displays the contents in new browser window. To read the existing UOW as server, logon parameters **Broker**, **Service**, **User ID** and **Token** are required.
Save	Saves the selection and changes.
Type: Name	Shows the name of referenced component. The name can be selected depending on the previously selected pipeline step type. **Note:** The selected component is referenced by name. After renaming the referenced component, this value must also be changed. Otherwise, this pipeline step will not be able to display any information about the referenced component.
Hide	Select **hide=no** to show this pipeline step. With **hide=yes** the step is hidden.

The following sections (**Inbound** and **Outbound**) explain 2 examples of pipelines. Display these and use them as templates to create your own.

The IDoc Inbound Pipeline

The IDoc Inbound pipeline is created on the Setup Wizard IDoc XML Gateway (*http://YourGateway:8080/sapr3gateway/manager/setupWizardIDocXMLGW*) with all steps. Use the **Show** link on the starting pipeline page to display the status of this pipeline.

Pipeline: IDoc Inbound Start Autorefresh System Log		Operation	Status
Queue	localhost@QUEUE/INBOUND-IDOC-XML/TEST		processed=56
Attach Manager	IDocService	Start Simulate	
Service	localhost@RPC/R3RFCD/CALLNAT		active server=4, requests=0, peding=0, peding high=0
Running Task	Rpc2RfcServerDev	Stop	Task has started

Use Natural or COBOL to create an IDoc document for this pipeline. The development section **Natural IDoc Client** (or for **COBOL**) describes all of the steps. After running the Natural IDoc PMQ client, you will see a message in the first pipeline step as accepted. Start the other pipeline steps (Attach Manager and Rpc2Rfc kernel) to deliver the document.

Note: The SAP R/3 application server must handle the IDoc type that is created by a partner. Please read the section to **configure the partner for a specific IDoc type**.

The `IDocService` **Attach Manager** receives a message from the EntireX Broker if one UOW is created. Now, the associated stylesheet `XMLToIDoc_FromPMQ` will be called. To display or change the following configuration, use the link with the name of the attach manager in pipeline view.

Service:	QUEUE/INBOUND-IDOC-XML/INT	
	URL:	XSLT_URL/XMLToIDoc_FromPMQ?rpc-service=RPC/R3RFCI/CALLNAT
	Log Response:	stdout
	Sleep Interval between 2 Events:	
	Synchronize parallel Requests:	false
	Simulate Delete	

The stylesheet performs the transformation and delivers the IDoc to SAP. For the delivery process, the **Rpc2Rfc kernel** is used. The function IDOC_INBOUND_ASYNCHRONOUS is called in transactional RFC (tRFC) mode. To call the Rpc2Rfc kernel service, the following parameters can be passed to the stylesheet.

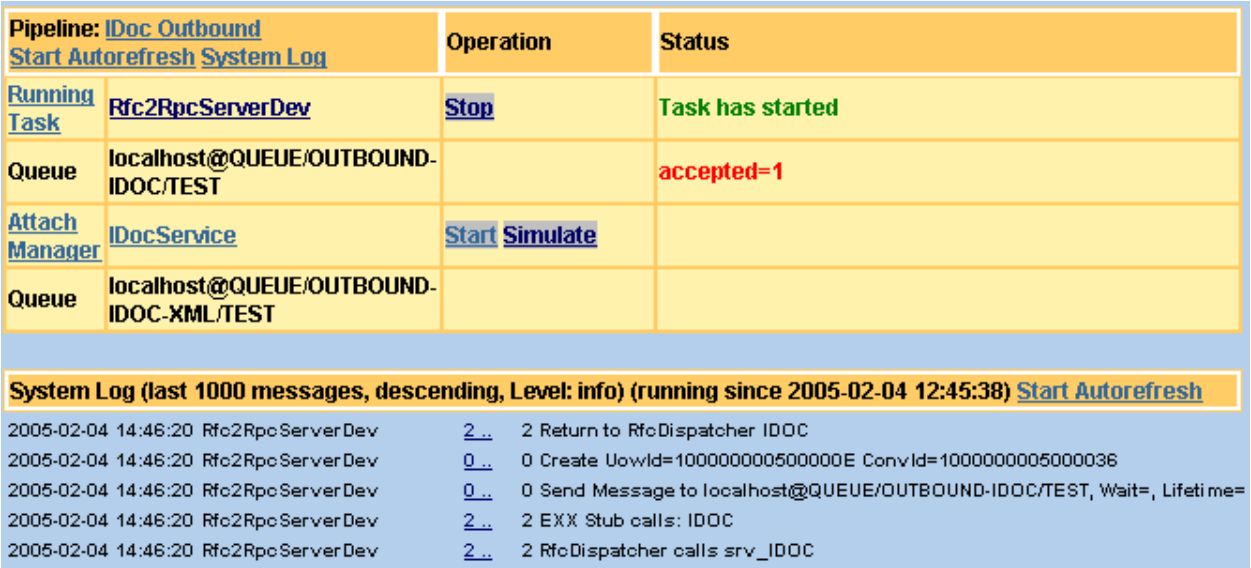
Parameter	Description
rpc-broker	Forwards the request to this broker
rpc-service	Sends the RPC request to this CLASS/SERVER/SERVICE.
rpc-userid	Connects with this user ID as client.
rpc-password	The EntireX security password for logon with user ID.

Trace Inbound Processing

1. The **System Log** contains information about the **Rpc2Rfc** and the result of `XMLToIDoc_FromPMQ` stylesheet processing. The delivery process is successful if there is a tRFC transaction ID logged.
2. To show the incoming IDoc in the SAP application server, start the transaction `we02`.
3. To trace the incoming tRFC requests, use transaction sm58.

The IDoc Outbound Pipeline

The IDoc Inbound pipeline is created on the Setup Wizard IDoc XML Gateway (*http://YourGateway:8080/sapr3gateway/manager/setupWizardIDocXMLGW*) with all of the steps. Use the **Show** link when starting a pipeline page to display the status of this pipeline. The following example displays one UOW in the pipeline



Pipeline: IDoc Outbound Start Autorefresh System Log		Operation	Status
Running Task	Rfc2RpcServerDev	Stop	Task has started
Queue	localhost@QUEUE/OUTBOUND-IDOC/TEST		accepted=1
Attach Manager	IDocService	Start Simulate	
Queue	localhost@QUEUE/OUTBOUND-IDOC-XML/TEST		

System Log (last 1000 messages, descending, Level: info) (running since 2005-02-04 12:45:38) Start Autorefresh

```
2005-02-04 14:46:20 Rfc2RpcServerDev   2 ..   2 Return to RfcDispatcher IDOC
2005-02-04 14:46:20 Rfc2RpcServerDev   0 ..   0 Create UowId=100000000500000E ConvId=1000000005000036
2005-02-04 14:46:20 Rfc2RpcServerDev   0 ..   0 Send Message to localhost@QUEUE/OUTBOUND-IDOC/TEST, Wait=, Lifetime=
2005-02-04 14:46:20 Rfc2RpcServerDev   2 ..   2 EXX Stub calls: IDOC
2005-02-04 14:46:20 Rfc2RpcServerDev   2 ..   2 RfcDispatcher calls srv_IDOC
```

Use the **Start** and **Simulate** links of the `IDocService` Attach Manager to transform plain IDoc format to XML. After this pipeline step, the IDoc is forwarded to the `QUEUE/OUTBOUND-IDOC-XML/TEST` queue.

To deliver the XML IDoc to your application, change the stylesheet which is called in Attach Manager `IDocService` for the service `QUEUE/OUTBOUND-IDOC-XML/TEST`. Depending on the stylesheet, there are additional parameters. Add these parameters as HTTP-GET parameters to the configuration of `IDocService`.

Available Stylesheet	Description
PMQdoc_toRPC	Reads the XML document from queue and forwards it to an HTTP XML-RPC server. Adds the `rpc-broker` and `rpc-service` parameters for the EntireX XML-RPC server. To develop an XML-RPC server for receiving XML documents and calling RPC subprograms, use the **Outbound XML-RPC Development** section.
PMQdoc_toMail	Reads the XML document from queue and sends the contents as mail. Adds the `to` parameter for the receiver's e-mail address.
PMQdoc_toFile	Reads the XML document and saves the contents as a file. The filename is created from the UOW ID. Adds the `path` parameter for the target directory.
PMQdoc_toPMQ	Reads the XML document and saves the contents in the next PMQ. Addresses the receiver PMQ with the `nextBroker` (the Broker IP/DNS) and `nextService` (CLASS/SERVER/SERVICE) parameters.
PMQdoc_toXBD	Reads the XML document and sends the contents to the Service Orchestrator. The receiver is a Mediator sequence. To address the sequence, add the HTTP parameter `href` for the Mediator web client and `xbd.sequence.uri` for calling the sequence.

All `PMQdoc` stylesheets include a customer stylesheet for your own transformations. The name of this stylesheet is *XMLIDoc_ToCustom.xsl* and does not perform transformation. The contents of PMQ are delivered 1:1 to the target RPC, mail or file. Add custom transformations to this stylesheet to receive IDocs in your business application.

Setup UOW Controller

The UOW controller checks all accepted units of work (UOWs) in one webMethods EntireX Broker (see the section **Using Persistence and Units of Work** in the webMethods EntireX documentation). If forward processing has stopped for one UOW, the next pipeline step will be searched. The next pipeline step will be called to restart pipeline processing.

For example, at the end of an outbound pipeline, the pipeline consumer must send a message to an e-mail system. The consumer performs a backout if an error is detected. The next UOW controller process restarts the pipeline consumer to deliver the message.

Create a **Scheduler Timer Task** to start the UOW controller within a certain period of time. Use the URL *SM_URL/UOWcontroller* in the field **Command**. Optionally, the following parameters can also be passed

HTTP-Get Parameter	Description
broker	The broker IP or DNS address.
deliveryCountLimit	The default is 1. It performs only UOWs with a delivery count of 1 or higher. Refer to the section **Using Persistence and Units of Work** in the webMethods EntireX documentation for more details.

Tip: Create two **Scheduler Timer Tasks**. One timer task works with `deliveryCountLimit=1` in a short period and the second with `deliveryCountLimit=0` in a 1-day period.

If you call the UOW controller from the menu, you will be prompted for the Broker IP or DNS address.

Using Pipeline View

The pipelines already defined allow you to control the business message flow. All of the pipeline steps defined for a pipeline are displayed on a page. To display this page, select the pipeline with the **Show** button on the **pipeline page**. On the selected pipeline, there are links to start the next activities.

Pipeline: IDoc Exchange Start Autorefresh System Log Refresh		Operation	Status
IDoc Inbound		Show	
Queue	localhost@QUEUE/INBOUND-IDOC-XML/TEST		
Attach Manager	IDocService	Stop Cancel Simulate	Has started
Service	localhost@RPC/R3RFCD/CALLNAT		Server active=2, Requests=0, Pending=0, Pending high=0, Conversation active=0, Conversation high=0, Long Buffer active=0, Long Buffer high=0, Short Buffer active=0, Short Buffer high=0
Running Task	Rpc2RfcServerDev	Stop	Task has started
IDoc Outbound		Show	
Running Task	Rfc2RpcServerDev	Stop	Task has started
Queue	localhost@QUEUE/OUTBOUND-IDOC/TEST		
Attach Manager	IDocService	Stop Cancel Simulate	Has started
Queue	localhost@QUEUE/OUTBOUND-IDOC-XML/TEST		
Attach Manager	IDocService	Stop Cancel Simulate	Has started

This example shows the inbound and outbound in one parent **IDoc Exchange** pipeline view. The children are indented.

Pipeline: ...

- The link of the selected pipeline shows the pipeline definition with all steps.
- **Start Autorefresh**, this pipeline view is automaticly refreshed. The refresh time is set on the **System Constants page**. Click this button to toggle this function, (to start and stop Autorefresh).
- **System Log** shows and refreshes the pipeline view additional with the **System Log**.
- The **Show** button displays only this pipeline.

The Pipeline Name

The pipeline title name is displayed if the parent definition contains a child pipeline step. This link shows the pipeline definition. The **Show** button displays only this pipeline on the next page. You can use this button to drill down to your information and display the page more quickly.

Attach Manager

- The **Attach Manager** is a working thread that waits for newly created UOWs and starts a stylesheet transformation process.
- The first link shows all attach managers. The selected attach manager displays its own definition. The **Start** button starts the attach manager. Once an attach manager has been started, this button is toggled to **Stop**. If you stop an attach manager, pipeline processing also stops.
- The **Simulate** button creates an event to start the working process. The status of the attach manager is changed and this is displayed in the status column. To show the result of processing, use the **System Log** button in the head of the pipeline view. The attach manager has written the result with its name into the log. The **Simulate** button allows you to process one UOW in the queue to the next pipeline step.

Pipeline: IDoc Inbound Start Autorefresh System Log Refresh		Operation	Status
Queue	localhost@QUEUE/INBOUND-IDOC-XML/TEST		
Attach Manager	IDocService	Stop Cancel Simulate	Simulate event is finished
Service	localhost@RPC/R3RFCD/CALLNAT		Server active=2, Requests=8, Pending=0, Pending high=1, Conversation active=0, Conversation high=1, Long Buffer active=0, Long Buffer high=10, Short Buffer active=0, Short Buffer high=0
Running Task	Rpc2RfcServerDev	Stop	Task has started

System Log (last 10000 messages, descending, Level: info) (running since 2005-03-07 09:24:34) Start Autorefresh

2005-04-20 14:07:53	IDocService / 10.22.19.42	<?x..	<?xml version="1.0" encoding="ISO-8859-1"?> <xmlToIDoc><open><har
2005-04-20 14:07:53	IDocService / 10.22.19.42	Sho..	Show next line the response stream ...
2005-04-20 14:07:53	IDocService / 10.22.19.42	HTT..	HTTP response code = 200, message = OK
2005-04-20 14:07:53	Rpc2RfcServerDev	[5..	[5] (0) Function exit
2005-04-20 14:07:53	Rpc2RfcServerDev	[5..	[5] (2) Function 'IDOCINBD' is called with operation=64
2005-04-20 14:07:52	Rpc2RfcServerDev	[4..	[4] (2) Function exit
2005-04-20 14:07:52	Rpc2RfcServerDev	[4..	[4] (0) Function 'IDOCINBD' is called with operation=3
2005-04-20 14:07:52	IDocService / 10.22.19.42	Att..	Attach Manager performs HTTP attach command = http://localhost:8080/sapr3idocxmlgateway/transformer/XMLToIDoc_Fror broker=localhost&service=QUEUE/INBOUND-IDOC-XML/TEST&userid=ID
2005-04-20 14:07:52	IDocService / 10.22.19.42	Att..	Attach Manager is simulating attach request.

Running Task

- The **Running Task** button displays the status of **all tasks**. The link with specific task name displays the parameters.
- The operation coloumn contains the **Start** and **Stop** button. Depending of the status, the **Start** or **Stop** button is available and it is toggled once it has been clicked.

Timer Task

- The **timer task** creates a processing event within a period of time and/or at a specific time. To show the configuration parameter, press the link with the name of the timer task.
- The **Start** button starts the timer task. It toggles to the Stop button once it has been clicked. At a specific time or period of time the result of a performed event is stored in an internal document.

- The **Simulate** button creates the processing event immediately. The result is saved in an internal document.
- To display the result, press the **Show** button in the status coloumn. If an exception has occurred during the processing event, an **Exception** button will become available.

Queue and Service

The **Queue** and **Service** rows display EntireX Broker resource information used by the pipeline messages.

Outbound XML-RPC Development

This development environment allows you to receive IDoc documents as a pipeline message consumer. The goal is to receive the IDoc document with the business parameter in a sub-program. The sub-program is called at runtime from an EntireX RPC server. The following development steps are necessary.

Deliver IDoc to RPC Server

1. Create and setup the **Outbound XML-RPC Development** with the Setup Wizard on **Help** and **Setup Wizard IDoc XML Gateway**, if this has not already been done. After creation, the development environment exists on the **Development** page.
2. Create the **IDoc type** information in IDoc XML gateway.
3. Select **Minimum IDL** to create the IDL from the **IDoc type** information. Copy the content of the newly created windows to your clipboard.
4. Go to the **Outbound XML-RPC Development** on the developer page, choose **Edit IDL** and copy the contents of the clipboard to the end of the existing IDL. Check if the new contents already exist.
5. Edit and expand the IDL.

   - Change the program name if your called sub-programm has another name for its own naming convention. Delete the alias name if you only have an 8-character-long name for the sub-program.
   - Delete the `EDI_DC40` group. These fields and information are not necessary in the called sub-program.
   - Add a group for response message and code. This will allow error handling if the IDoc document cannot be delivered. Set the direction of this group to `OUT`.

   In summary to make all of the changes, the following IDL can used as an example:

```
Program 'IDOCRCVN' Is       /*  'DEBCOR'
  Define Data Parameter
    1 MESSAGE  Out
      2 NO (N4)
      2 TEXT (A60)
    1 E1KNA1C  In /* Core Master Kundenstamm Grunddaten (KNA1)
      2 MSGFN (A3)        /* Function
      2 KUNNR (A10)        /* Customer number
      2 ANRED (A15)        /* Title
      ....
      2 SPRAS_ISO (A2)        /* Language according to ISO 639
  End-Define
```

6 Click **Generate Source** to generate an XMM mapping file. The EntireX Workbench is called in batch mode. The mapping file is copied to the IDoc XML Gateway. To change the destination directory, go to **Configuration**, **Development**, and select this development environment. The XMM mapping file is used by an EntireX servlet adapter to map an XML document to an RPC call.

7 To activate the new XMM file, restart IDoc XML gateway. For example, use the Tomcat manager.

8 Develop a sub-program (e.g. Natural) with the name and parameter data area which is defined in the IDL (step 5 above).

9 Start and check the RPC server which is to call the developed sub-program.

10 Change the **attach manager** configuration to send the outbound IDoc as XML. Set the `PMQdoc_toRPC` stylesheet and the RPC HTTP-Get parameter for the XML-RPC adapter.

HTTP Get Parameter	Description
rpc-broker	Broker IP-address of the called sub-program.
rpc-service	Sevice (CLASS/SERVER/SERVICE) of the called sub-program.

For example, use

```
XSLT_URL/PMQdoc_toRPC?rpc-broker=localhost&rpc-service=RPC/SRV1/CALLNAT
```

as **URL** on service `OUTBOUND-IDOC-XML`.

`XSLT_URL` is replaced at runtime by the value defined in **System Constants**. To activate the change, stop and restart the Attach Manager.

11 Depending on step 5, the next section describes a transformation from IDoc to RPC.

Develop IDoc to RPC Transformation

As default, the IDoc is transformed to XML and this XML document calls the RPC server. To transform between the IDoc-XML document and the required XML-RPC document, it is possible to add XSL transformation statements.

1. To transform between the two XML documents, the `XMLIDoc_ToCustom.xsl` stylesheet is called from IDoc XML Gateway at runtime. To add your own transformation, go to **Tools**, **Files** and edit the file **XML IDoc to Custom Transformation**, or start an editor with this file in the root directory of IDoc XML Gateway.
2. Add an `xsl:template` that matches if the IDoc type is transformed. Use the attribute `priority` to overwrite the default.

```
<xsl:template match="DEBCOR01" priority="1">
  ....
</xsl:template>
```

   In this example the `DEBCOR01` IDoc type matches.

3. To this template, add rules to transform the IDoc type name to the sub-program name.

```
<xsl:template match="DEBCOR01" priority="1">
   <IDOCRCVN>
      ....
   </IDOCRCVN>
</xsl:template>
```

4. Copy 1:1 all IDoc segments to the XML-RPC document.

```
<xsl:template match="DEBCOR01" priority="1">
   <IDOCRCVN>
      <xsl:copy-of select="//E1KNA1C"/>
   </IDOCRCVN>
</xsl:template>
```

5. The called sub-program can determine at runtime if the document has been successfully processed. The `out` parameter replies with this information. The response code `true` commits the IDoc document as a UOW and `false` backs out. It this case, the same document can also be delivered later. To reply to the response, add an `xsl:template` with mode `doEvaluateResponseCode`.

```
<xsl:template match="*" mode="doEvaluateResponseCode">
   ....
</xsl:template>
```

6 The RPC response document is passed to this template. It is now possible to evaluate whether to commit the UOW (return `true`) or to back it out (return `false`).

```
<xsl:template match="*" mode="doEvaluateResponseCode">
   <xsl:choose>
      <xsl:when test="//MESSAGE/NO = '4'">
         <!-- Adabas Record successfully stored -->
         <xsl:text>true</xsl:text>
      </xsl:when>

      <xsl:when test="//MESSAGE/NO = '5'">
         <!-- Adabas Record successfully updated -->
         <xsl:text>true</xsl:text>
      </xsl:when>

      <xsl:otherwise>
         <xsl:text>false</xsl:text>
      </xsl:otherwise>
   </xsl:choose>
</xsl:template>
```

Note: The XPath expression contains the field names of IDL in the above example.

7 To restart the delivery process after an UOW backout, implement a **UOW Controller**.

8 In case of error (backout of UOW), the response message of the called sub-program can be logged. Add an `xsl:template` with mode `doEvaluateResponseMessage`. The RPC response document is passed to this template.

```
<xsl:template match="*" mode="doEvaluateResponseMessage">
   <xsl:value-of select="//MESSAGE/TEXT"/>
</xsl:template>
```

28 Natural IDoc Client

This section describes all of the development steps required to create a Natural IDoc client. This client creates messages for the **IDoc inbound pipeline**. The same interface is provided for **COBOL**.

To setup development

- The development page (*http://YourGateway:8080/sapr3idocxmlgateway/transformer/IDocType_ToFile* or menu item **Development**) provides an IDL text editor and the associated batch commands to develop a Natural PMQ Client. To set up this page, use the setup IDoc XML Gateway wizard on *http://YourGateway:8080/sapr3gateway/manager/setupWizardIDocXMLGW#11* and perform step 11 **Create Natural IDoc PMQ Development Environment**.

To create a Natural IDoc Client

1. Create the **IDoc type** information for your client on *http://YourGateway:8080/sapr3idocxmlgateway/transformer/IDocType_ToFile* or use the menu items **Go** , **IDoc XML Gateway** and **Development**.
2. Use the **Minimum IDL** link to open the IDL for a Natural IDoc client in a new browser window.

   The IDoc type DEBCOR01 returns the following IDL:

```
Program 'DEBCOR01':'DEBCOR01' Is
  Define Data Parameter
    1 EDI_DC40 IN /* IDoc control record
      2 MESTYP  (A30)   /* Message type
      2 SNDPRT  (A2)    /* Partner type of sender
      2 SNDPRN  (A10)   /* Partner number of sender
    1 E1KNA1C IN   /*   (Must=X, OccMin=1, OccMax=9999, GMust=, GOccMin=0, ↩
GOccMax=0) Core master customer master basic data (KNA1)
      2 MSGFN (A3)        /* Function
      2 KUNNR (A10)        /* Customer number
      2 ANRED (A15)        /* Title
```

```
    2 KTOKD (A4)        /* Customer Account Group
    2 LOEVM (A1)        /* Central Deletion Flag for Master Record
    2 NAME1 (A35)        /* Name 1
    2 NAME2 (A35)        /* Name 2
    2 ORT01 (A35)        /* City
    2 ORT02 (A35)        /* District
    2 PFACH (A10)        /* P.O. Box
    2 PSTL2 (A10)        /* P.O. Box postal code
    2 PSTLZ (A10)        /* Postal Code
    2 SORTL (A10)        /* Sort field
    2 STRAS (A30)        /* House number and street
    2 LAND1 (A3)        /* Country key
    2 SPRAS (A1)        /* Language key
    2 SPRAS_ISO (A2)        /* Language according to ISO 639
End-Define
```

3 Mark and copy the contents of the browser window to your clipboard.

4 Start the developer editor for Natural PMQ client on the development page *http://YourGateway:8080/sapr3gateway/manager/dev*. This development environment is created by the Setup Wizard IDoc XML Gateway (*http://YourGateway:8080/sapr3gateway/manager/setupWizardIDocXMLGW*).

5 Paste the clipboard contents to the end of the edit IDL.

Development of: Natural IDoc PMQ Client

List IDL

Edit IDL

Generate Source

Result of generation Process

Download SYSTRANS File

Edit IDL for: Natural IDoc PMQ Client

```
Library 'IDOC' Is

Program 'DEBCOR':'DEBCOR01' Is
  Define Data Parameter
    1 EDI_DC40 IN /* IDoc control record
      2 MESTYP  (A30)    /* Message type
      2 SNDPRT  (A2)     /* Partner type of sender
      2 SNDPRN  (A10)    /* Partner number of sender
    1 E1KNA1C IN   /*   (Must=X, OccMin=1, OccMax=9999, GMust=, GOccMin=0,
GOccMax=0) Core Master Kundenstamm Grunddaten (KNA1)
      2 MSGFN (A3)         /* Funktion
      2 KUNNR (A10)         /* Debitorennummer
      2 ANRED (A15)         /* Anrede
      2 KTOKD (A4)         /* Kontengruppe Debitor
      2 LOEVM (A1)         /* Zentrale Löschvormerkung für Stammsatz
      2 NAME1 (A35)         /* Name 1
      2 NAME2 (A35)         /* Name 2
      2 ORT01 (A35)         /* Ort
      2 ORT02 (A35)         /* Ortsteil
      2 PFACH (A10)         /* Postfach
      2 PSTL2 (A10)         /* Postleitzahl des Postfachs
      2 PSTLZ (A10)         /* Postleitzahl
      2 SORTL (A10)         /* Sortierfeld
      2 STRAS (A30)         /* Straße und Hausnummer
      2 LAND1 (A3)         /* Länderschlüssel
      2 SPRAS (A1)         /* Sprachenschlüssel
      2 SPRAS_ISO (A2)         /* Sprache nach ISO 639
  End-Define
```

Save

6 Change the 8-character short name of the program statement. In the `DEBCOR01` example, you can change it to `DEBCOR`. The starting test program will have the name `DEBCORP`.

7 Click the **Save** button.

8 Click **Generate Source**.

9 Transport the generated SYSTRANS file to your Natural development environment using the function **Download SYSTRANS file**.

10 Start the SYSTRANS utility in your Natural development environment and import the file.

11 The target Natural library is the library that is used in the library statement of IDL. In this example: `IDOC`.

12 Compile all of the sources with the `catall` command.

13 Start the program to edit the example with the same IDoc type name. In this example: `DEBCORP`. See **inbound configuration** of partner profile for more parameter values.

```
* ----------------------------------------------------------------------------
*
* Test program for sending an XML document in a unit of work
* This source is automatically generated from IDL
*
* ----------------------------------------------------------------------------
DEFINE DATA  /* DEBCOR01
  LOCAL USING PMQINFOA
  LOCAL USING DEBCOR0A
END-DEFINE
*
* DO NEXT INIT CALL ONCE IN THE SESSION
* AND SET THE QUEUE CONNECTION PARAMETER
CALLNAT 'PMQINIT' PMQ-SERVICE PMQ-RETURN
*
* DO NEXT OPEN CALL FOR EACH UNIT OF WORK
CALLNAT 'PMQOPEN' PMQ-SERVICE PMQ-RETURN
...
```

Set all fields with the values that are necessary for your client

```
...
* SET VALUES
EDI_DC40.MESTYP := 'DEBCOR' /* TYPE = A(30)
EDI_DC40.SNDPRT := 'LS' /* TYPE = A(2)
EDI_DC40.SNDPRN := 'SRG' /* TYPE = A(10)
E1KNA1C.MSGFN := '005' /* TYPE = A(3)
...
```

Make no changes to document sending and error handling:

```
* WRITE GROUP EDI_DC40 INTO STREAM
CALLNAT 'EDI_DC40' EDI_DC40
*
* WRITE GROUP E1KNA1C INTO STREAM
CALLNAT 'E1KNA1C' E1KNA1C
*
* COMMIT AND CLOSE STREAM
CALLNAT 'PMQCMMT'
* ASK FOR STREAM INFORMATION
CALLNAT 'PMQINFO' PMQ-SERVICE PMQ-RETURN
*
WRITE 'OK     :' PMQ-RETURN.OK
```

```
IF PMQ-RETURN.OK THEN
  WRITE 'CONV-ID:' PMQ-RETURN.CONV-ID
  WRITE 'UOW-ID :' PMQ-RETURN.UOW-ID
ELSE
  WRITE 'CODE   :' PMQ-RETURN.CODE
  WRITE 'MESSAGE:' PMQ-RETURN.MESSAGE
END-IF
END                    /* END OF DEBCOR
```

14 Set the connection parameter `BROKER-ID` to the **inbound pipeline** in the generated subprogram `PMQINIT`.

```
...
*
* SET QUEUE CONNECTION PARAMETER
BROKER-ID    := 'localhost'
SERVER-NAME  := 'INBOUND-IDOC-XML'
SERVER-CLASS := 'QUEUE'
SERVICE      := 'TEST'
*
USERID   := *INIT-USER
PASSWORD := ' '
*
...
```

To set the connection parameter on generation process, the following parameter can be set with `-D KEY=VALUE` syntax in the make command.

Key Parameter	Value Description	Default Value
SUPPRESS	The value `PMQINIT` suppresses the generation of this subprogram in the SYSTRANS file.	
BROKER	Sets the value for Broker ID.	localhost
CLASS	Sets the value for class name.	QUEUE
SERVER	Sets the value for server name.	INBOUND-IDOC-XML
SERVICE	Sets the value for service name.	TEST
USERID	Log on with this user ID as client .	The Natural system variable `*INIT-USER`
PASSWORD	User ID's password with EntireX Security.	

To change the make command for setting parameters, go to **Configuration**, **Development**, **Natural PMQ Client**, **Make startup script** menu item and edit the command line in the opened text editor.

```
...bin/erxidl -D BROKER=ETB001 -D FILE=PMQ.txt -t NatPMQClient.tpl $IDL
```

This example (on UNIX) sets the Broker ID to value `ETB001` and overwrites the default.

15 Run the test program. The test program writes the conversation ID and UOW ID to the output.

16 Go to the **inbound pipeline** and display the processing status.

17 Start the transaction `we02` to display the inbound IDoc in the SAP application server. Make sure that the **inbound partner profile** is configured.

To change the IDoc interface

There are cases in which the IDoc interface *may* be changed and cases in which it *must* be changed. Use the following rules during the development steps to optimize the IDoc interface.

1 The library name in IDL is the target Natural library. The SYSTRANS fills the Natural objects in this library.

2 The 8-character program name is used to generate the test program (postfix `P`), the PDA (postfix `A`), and the subprogram for creating the IDoc control record.

3 The alias of the 8-character program name is used for the XML main tag, which is the IDoc type name. Do not change this name. Do not change the group of field names.

4 Fields that are not needed can be deleted in IDL.

5 Groups that are not needed (=IDoc segment) can be deleted in IDL or the associated subprogram not called in the PMQ interface. Call a subprogram multiple, if this IDoc segment has multiple occurrences.

6 Do not change the calling sequence of subprograms.

7 The name of the segment is the sub-program name, if this name is less than or equal to 8 characters. The IDoc type determines whether the segment name is longer than the sub-program name. In this case, the first 6 characters become the IDoc type name and the last 2 are an index.

Alternative architecture

If the changes are too complex to use the IDoc interface from your existing application, consider the following implementation architecture.

1 Create your own IDL for exporting data from the business system.

2 Generate, import and call the PMQ interface.

3 Create a pipeline step to call the stylesheet you have created. This stylesheet transforms the XML document generated from your PMQ Interface to the IDoc XML structure. To get the schema of the IDoc structure, use the SAP R/3 interface repository link on the overview page.

4 To create this pipeline step, use the **PMQdoc_toPMQ** stylesheet. This stylesheet calls the stylesheet you created in the previous step.

5 Post the IDoc XML document into the `QUEUE/INBOUND-IDOC-XML/TEST` queue.

29 COBOL IDoc Client

This section describes all of the development steps required to create a COBOL IDoc client. This client creates messages for the **IDoc inbound pipeline**. The same interface is provided for **Natural**.

To setup the Development

- The development page (*http://YourGateway:8080/sapr3idocxmlgateway/transformer/IDocType_ToFile* or menu item **Development**) provides an IDL text editor and the associated batch commands to develop a COBOL PMQ Client. To setup this page, use the setup IDoc XML Gateway wizard on *http://YourGateway:8080/sapr3gateway/manager/setupWizardIDocXMLGW#12* and perform step 12 **Create COBOL IDoc PMQ Development Environment**.

To create a COBOL IDoc Client

1. Create the **IDoc type** information for your client on *http://YourGateway:8080/sapr3idocxmlgateway/transformer/IDocType_ToFile* or use the menu items **Go** , **IDoc XML Gateway** and **Development**.
2. Use the **Minimum IDL** link to open the IDL for a COBOL IDoc client in a new browser window.

   The IDoc type DEBCOR01 returns the following IDL:

```
Program 'DEBCOR01':'DEBCOR01' Is
  Define Data Parameter
    1 EDI_DC40 IN /* IDoc control record
      2 MESTYP  (A30)   /* Message type
      2 SNDPRT  (A2)    /* Partner type of sender
      2 SNDPRN  (A10)   /* Partner number of sender
    1 E1KNA1C IN   /*   (Must=X, OccMin=1, OccMax=9999, GMust=, GOccMin=0, ↩
GOccMax=0) Core master customer master basic data (KNA1)
      2 MSGFN (A3)        /* Function
      2 KUNNR (A10)        /* Customer number
```

```
    2 ANRED (A15)        /* Title
    2 KTOKD (A4)        /* Customer Account Group
    2 LOEVM (A1)        /* Central Deletion Flag for Master Record
    2 NAME1 (A35)        /* Name 1
    2 NAME2 (A35)        /* Name 2
    2 ORT01 (A35)        /* City
    2 ORT02 (A35)        /* District
    2 PFACH (A10)        /* P.O. Box
    2 PSTL2 (A10)        /* P.O. Box postal code
    2 PSTLZ (A10)        /* Postal Code
    2 SORTL (A10)        /* Sort field
    2 STRAS (A30)        /* House number and street
    2 LAND1 (A3)        /* Country key
    2 SPRAS (A1)        /* Language key
    2 SPRAS_ISO (A2)        /* Language according to ISO 639
End-Define
```

3 Mark and copy the contents of the browser window to your clipboard.

4 Start the developer editor for COBOL PMQ client on the development page *http://YourGateway:8080/sapr3gateway/manager/dev*. This development environment is created by the Setup Wizard IDoc XML Gateway (*http://YourGateway:8080/sapr3gateway/manager/setupWizardIDocXMLGW*).

5 Paste the clipboard contents to the end of the edit IDL.

Development of: Cobol IDoc PMQ Client

List IDL

Edit IDL

Generate Source

Result of generation Process

Cobol Source

Edit IDL for: Cobol IDoc PMQ Client

```
Library 'IDOC' Is

Program 'DEBCOR':'DEBCOR01' Is
  Define Data Parameter
    1 EDI_DC40 IN /* IDoc control record
      2 MESTYP  (A30)    /* Message type
      2 SNDPRT  (A2)     /* Partner type of sender
      2 SNDPRN  (A10)    /* Partner number of sender
    1 E1KNA1C IN   /*   (Must=X, OccMin=1, OccMax=9999, GMust=, GOccMin=0,
GOccMax=0) Core master customer master basic data (KNA1)
      2 MSGFN (A3)         /* Function
      2 KUNNR (A10)         /* Customer number
      2 ANRED (A15)         /* Title
      2 KTOKD (A4)         /* Customer Account Group
      2 LOEVM (A1)         /* Central Deletion Flag for Master Record
      2 NAME1 (A35)         /* Name 1
      2 NAME2 (A35)         /* Name 2
      2 ORT01 (A35)         /* City
      2 ORT02 (A35)         /* District
      2 PFACH (A10)         /* P.O. Box
      2 PSTL2 (A10)         /* P.O. Box postal code
      2 PSTLZ (A10)         /* Postal Code
      2 SORTL (A10)         /* Sort field
      2 STRAS (A30)         /* House number and street
      2 LAND1 (A3)         /* Country key
      2 SPRAS (A1)         /* Language key
      2 SPRAS_ISO (A2)         /* Language according to ISO 639
  End-Define
```

Save

6 Change the 8-character short name of the program statement. In the `DEBCOR01` example, you can change it to `DEBCOR`. The starting test program will have the name `DEBCORP`.

7 Click the **Save** button.

8 Click **Generate Source**. Return code 0 indicates a successful generation process.

9 Use **COBOL Source** to browse the generated files. All subprograms of the PMQ interface have the prefix `PMQ` and for all **IDoc types** the same. The other file names depend on the name of the IDoc type. Following files are generated, for example with DEBCOR as IDoc type:

File	Description
PMQBCKT	Disable the document delivery. Backout an allocated Unit of Work.
PMQCMMT	Commits and closes the created document as Unit of Work.
PMQINFO	Returns status and error information of current allocated Unit of Work.
PMQINIT	Initializes the PMW environment and set the connection parameter.
PMQOPEN	Starts a new document stream as Unit of Work.
PMQCOPY	Copy code for general PMQ parameter and status interface..
DEBCORP	Example program to deliver a complete debitor (DEBCOR) as document.
DEBCOR	Writes IDoc header information of DEBCOR in to the opened document stream.
E1KNA1C	Writes IDoc business data of a debitor in to the opened document stream.
DEBCORC	Copy code of DEBCOR business IDoc interface.

10 Transport the generated source code files to your COBOL development environment. When you press the **Generate Source** button, an FTP batch comand file (*ftp.bat* for Windows) or a shell script (*ftp.sh* for UNIX) is generated. This command script transports the generated source code files with FTP to your target file system. By default, the execution of this script is uncommended. To apply the transport, you need the parameter IP/DNS name of target host, FTP user ID, password and target directory (optional). As well, you can change the target COBOL enviromnent `BATCH` or `CICS`.

To set the parameter, go **Configuration**, **Development** and select **COBOL IDoc PMQ Client**. The **Make startup script** link starts an editor. The default script for Windows:

```
@echo off

rem Set target Cobol environment
set TARGET=BATCH
rem set TARGET=CICS

%EXXDIR%/bin/erxidl -DTARGET=%TARGET% -t CobolPMQClient.tpl %IDL%
%EXXDIR%/bin/erxidl -DSHELL=BAT       -t ftp.tpl            %IDL%

rem Upload Cobol source
rem ftp.bat host userid password directory
```

The default script for UNIX:

```
#!/bin/sh

PATH=/bin:/usr/bin:$PATH
export PATH

# Set Environment for SAG-Environment
echo Call $SAGENV
. $SAGENV

# Set target Cobol environment
TARGET=BATCH
# TARGET=CICS

$EXXDIR/$EXXVERS/bin/erxidl -DTARGET=$TARGET -t CobolPMQClient.tpl $IDL
$EXXDIR/$EXXVERS/bin/erxidl -DSHELL=SH       -t ftp.tpl            $IDL

# Upload Cobol source
# ./ftp.sh host userid password directory
```

11 Compile the uploaded sources in your COBOL development environment. The first time, you must compile the subprogram with prefix `PMQ`. The next time, only the new programs and subprograms of the IDoc type are required for compilation.

12 Start the program to edit the example with the same IDoc type name. In this example: `DEBCORP`. See **inbound configuration** of partner profile for more parameter values.

```
       ID DIVISION.
       PROGRAM-ID.   DEBCORP.

      *-----------------------------------------------------------------*
      * Copyright (c) 2004 by SAG Systemhaus GmbH. All rights reserved.
      * SAP R/3 Gateway
      * Version: 2.3.1
      * This program use the defined interface to create
      * a Unit of Work as document.
      *-----------------------------------------------------------------*
       ENVIRONMENT DIVISION.

       CONFIGURATION SECTION.
       SPECIAL-NAMES.
           CONSOLE IS MY-TERMINAL.

       DATA DIVISION.

      * --- WORKING ----------------------------------------------------
       WORKING-STORAGE SECTION.

      * PMQ STANDARD INTERFACE
       01  PMQ-SERVICE.
           02  PMQ-BROKER-ID             PIC X(32).
           02  PMQ-SERVER-CLASS          PIC X(32).
```

```
          02  PMQ-SERVER-NAME           PIC X(32).
          02  PMQ-SERVICE-NAME          PIC X(32).
          02  PMQ-USERID                PIC X(32).
          02  PMQ-PASSWORD              PIC X(32).
          02  PMQ-TOKEN                 PIC X(32).
      01  PMQ-RETURN.
          02  PMQ-OK                    PIC X(1).
          02  PMQ-MESSAGE               PIC X(40).
          02  PMQ-CODE                  PIC X(8).
          02  PMQ-CONV-ID               PIC X(16).
          02  PMQ-UOW-ID                PIC X(16).


     * DOCUMENT INTERFACE
      01 EDI-DC40.
          10 MESTYP-1   PIC X(30).
          10 SNDPRT-2   PIC X(2).
          10 SNDPRN-3   PIC X(10).
      01 E1KNA1C.
          10 MSGFN-4   PIC X(3).
          10 KUNNR-5   PIC X(10).
          10 ANRED-6   PIC X(15).
          10 KTOKD-7   PIC X(4).
          10 LOEVM-8   PIC X(1).
          10 NAME1-9   PIC X(35).
          10 NAME2-10   PIC X(35).
          10 ORT01-11   PIC X(35).
          10 ORT02-12   PIC X(35).
          10 PFACH-13   PIC X(10).
          10 PSTL2-14   PIC X(10).
          10 PSTLZ-15   PIC X(10).
          10 SORTL-16   PIC X(10).
          10 STRAS-17   PIC X(30).
          10 LAND1-18   PIC X(3).
          10 SPRAS-19   PIC X(1).
          10 SPRAS-ISO-20   PIC X(2).

      PROCEDURE DIVISION.

     * INIT PMQ INFO AND CONNECTION PARAMETER
          INITIALIZE PMQ-SERVICE, PMQ-RETURN.

     * INIT DOCUMENT INTERFACE
          INITIALIZE EDI-DC40.
          INITIALIZE E1KNA1C.

     * DO NEXT INIT CALL ONCE IN THE SESSION
          CALL 'PMQINIT' USING PMQ-SERVICE, PMQ-RETURN.

     * DO NEXT OPEN CALL FOR EACH UNIT OF WORK
          CALL 'PMQOPEN' USING PMQ-SERVICE, PMQ-RETURN.
...
```

Set all fields with the values that are necessary for your client

```
...
      * SET VALUES

      *  USING GROUP EDI-DC40
           MOVE 'DEBCOR' TO MESTYP-1.
           MOVE 'LS' TO SNDPRT-2.
           MOVE 'SRG' TO SNDPRN-3.

      *  USING GROUP E1KNA1C
           MOVE '005' TO MSGFN-4.
...
```

Make no changes to document sending and error handling:

```
      * WRITE 'EDI-DC40' GROUP INTO THE STREAM
           CALL 'DEBCOR' USING EDI-DC40.

      * WRITE 'E1KNA1C' GROUP INTO THE STREAM
           CALL 'E1KNA1C' USING E1KNA1C.

      * COMMIT AND CLOSE STREAM
           CALL 'PMQCMMT'.

      * ASK FOR STREAM INFORMATION
           CALL 'PMQINFO' USING PMQ-SERVICE, PMQ-RETURN.

      * CHECK UOW
           DISPLAY 'Send document to Broker: ' PMQ-BROKER-ID.
           IF PMQ-OK = 1
             DISPLAY 'CONV-ID:' PMQ-CONV-ID
             DISPLAY 'UOW-ID :' PMQ-UOW-ID
           ELSE
             DISPLAY 'CODE   :' PMQ-CODE
             DISPLAY 'MESSAGE:' PMQ-MESSAGE
           END-IF.

       MAIN-EXIT.
           STOP RUN.

       END PROGRAM DEBCORP.
```

13 Set the connection parameter `BROKER-ID` to the **inbound pipeline** in the generated subprogram `PMQINIT`.

```
...
       INIT-USER-DEFINED-PMQ-PARAM.

      * SET QUEUE CONNECTION PARAMETER
           MOVE 'localhost'          TO PMQ-BROKER-ID.
           MOVE 'INBOUND-IDOC-XML'   TO PMQ-SERVER-NAME.
           MOVE 'QUEUE'              TO PMQ-SERVER-CLASS.
           MOVE 'TEST'               TO PMQ-SERVICE-NAME.

           MOVE 'PMQ'                TO PMQ-USERID.
           MOVE ' '                  TO PMQ-PASSWORD.
...
```

To set the connection parameter on generation process, the following parameters can set with `-D KEY=VALUE` syntax in the make command.

Key Parameter	Value Description	Default Value
SUPPRESS	The value `PMQINIT` suppresses the generation of this subprogram.	
BROKER	Sets the value for Broker ID.	localhost
CLASS	Sets the value for class name.	QUEUE
SERVER	Sets the value for server name.	INBOUND-IDOC-XML
SERVICE	Sets the value for service name.	TEST
USERID	Log on with this user ID as client .	PMQ
TOKEN	Log on with this user ID and token as unique client.	
PASSWORD	User ID's password with EntireX Security.	

To change the make command for setting parameters, go to **Configuration**, **Development**, **Natural PMQ Client**, **Make startup script** menu item and edit the command line in the text editor.

```
...bin/erxidl -D BROKER=ETB001 -D FILE=PMQ.txt -t NatPMQClient.tpl $IDL
```

This example (on UNIX) sets the Broker ID to value `ETB001` and overwrites the default.

14 Run the test program. The test program writes the conversation ID and UOW ID to the output.

15 Go to the **inbound pipeline** and display the processing status.

16 Start the transaction `we02` to display the inbound IDoc in the SAP application server. Make sure that the **inbound partner profile** is configured.

To change the IDoc interface

There are cases in which the IDoc interface *may* be changed and cases in which it *must* be changed. Use the following rules during the development steps to optimize the IDoc interface.

1. The 8-character program name is used to generate the test program (postfix `P`), the copy code (postfix `C`), and the subprogram for creating the IDoc control record.
2. The alias of the 8-character program name is used for the XML main tag, which is the IDoc type name. Do not change this name. Do not change the group of field names.
3. Fields that are not needed can be deleted in IDL.
4. Groups that are not needed (=IDoc segment) can be deleted in IDL or the associated subprogram not called in the PMQ interface. Call a subprogram multiple, if this IDoc segment has multiple occurrences.
5. Do not change the calling sequence of subprograms.
6. The name of the segment is the sub-program name, if this name is less than or equal to 8 characters. The IDoc type determines whether the segment name is longer than the sub-program name. In this case, the first 6 characters become the IDoc type name and the last 2 are an index.

30 Software Development Kit

The Software Development Kit describes XSLT extensions to develop your own stylesheets for pipeline processing. There are 2 different functions in XSLT:

- XPath function
- element extension call.

The calling convention differs between the types. The XPath function requires parameters and the element extension requires attributes. In the function definition there is an example of how to call and use it. The XSLT function can distinguish between categories for

- sending and receiving UOWs,
- providing error and status information,
- handling UOW transactions,
- logging,
- exchanging documents via HTTP and
- getting system or HTTP-Get parameters.

The extension functions are described in detail:

Expanding xsl:stylesheet Header

To use the XSLT extension functions, the stylesheet header must be expanded.

```
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     version="1.0"
     xmlns:java="http://xml.apache.org/xslt/java"
     xmlns:io="XSLIOHelper"
     xmlns:properties="TransformerRequestProperties"
    extension-element-prefixes="io properties">
```

As convention the following sections use the definitions `xmlns:io` and `xmlns:properties` in function declarations.

Trace UOW function

To trace the communication to EntireX Broker, set the **JVM property** `XSLIOHelper.entirex.trace`. Use the value for this property defined on the page **ACI for Java.**

The trace output is written to the **System Log**.

Receiving UOW

Receives as server a UOW from the persistent message queue and returns an XML document. Use this function as XPath function.

Function Declaration:

```
io:receiveUOW       ( String brokerID, String service [, String userID [, String ↩
token [, String password ] ] ] )
io:receiveWithOldUOW( String brokerID, String service [, String userID [, String ↩
token [, String password ] ] ] )
```

The contents of the received UOW is required as XML document and passed through 1:1 to the receiver. The `WithOld` in the function name means that the server looks for and existing old conversation. If an error occurs, the returned XML document contains the error message. The error message starts with the `exception` XML tag.

Element Extension Example:

```
<xsl:variable name="item" select="io:receiveWithOldUOW( $broker, $service, $userid, ↩
'pmqToFile' )"/>
```

Stores the UOW as XML document in the `item` variable.

Parameter:

No.	Value	Description
1.	XPath expression should return a string.	Connection Broker ID as DNS name or IP address with port.
2.	XPath expression should return a string.	Service name in the format CLASS/SERVER/SERVICE
3.	XPath expression should return a string.	Connection user ID
4.	XPath expression should return a string.	Token for unique identification of this communication client.
5.	XPath expression should return a string.	Password for EntireX Broker Security.

Function Declaration:

```
io:receiveUOWText      ( String brokerID, String service [, String userID [, String ↩
token [, String password ] ] ] )
io:receiveWithOldUOWText( String brokerID, String service [, String userID [, String ↩
token [, String password ] ] ] )
```

The UOW is received as one document and tagged with the `token` XML tag.

Function Declaration:

```
io:receiveUOWLines       ( String brokerID, String service [, String userID [, ↩
String token [, String password ] ] ] )
io:receiveWithOldUOWLines( String brokerID, String service [, String userID [, ↩
String token [, String password ] ] ] )
io:receiveOldUOWLines( String brokerID, String service [, String userID [, String ↩
token [, String password ] ] ] )
```

The UOW can have many messages. Each message line is tagged with the `token` XML tag. Depending on function name, `io:receiveUOWLines()` queries only for new conversation, `io:receiveWithOldUOWLines()` queries for old and new, `io:receiveOldUOWLines()` queries only for old conversation. Old conversation is already reveived and opened with the same user ID and token parameter.

Sending UOW

Creates a UOW in the persistent message queue. This function should be used as an element extension call. The contents of an element extension are sent to the queue. The created UOW ID can be received with the `io:getUnitOfWorkID()` function. The `io:isOK()` call returns true on success or false on error.

Example:

```
<io:sendUOW
   broker="$broker"
   service="concat( 'QUEUE/OUTBOUND-IDOC-XML/', $environment )"
   user="$userid"
   token="'IDocToPMQ'"
   password="$password"
   mode="'client'"
   withOld="'false'"
   context="io:getUnitOfWorkID()">
        <xsl:copy-of select="$result"/>
</io:sendUOW>
```

The example sends the contents of the `$result` variable to the persistent message queue.

Attributes:

Name	Value	Description
broker	XPath expression should return a string.	Connection Broker ID as DNS name or IP address with port.
service	XPath expression should return a string.	Service name in format CLASS/SERVER/SERVICE
userid	XPath expression should return a string.	Connection user ID
token	XPath expression should return a string.	Token for identify this communication client unique.
password	XPath expression should return a string.	Password for EntireX Broker Security.
withOld	XPath expression should return 'true' or 'false'	Ask for an existing conversation and if true, append this. Otherwise, create a new conversation.
mode	XPath expression should return 'client' or 'server'	Creates this UOW as client (=sender) or as server (=receiver) to an existing UOW.
context	XPath expression should return a string.	Prints this context information to the UOW status.
encoding	XPath expression should return a string.	Defines the encoding of sending data.

Getting Status Information

The following set of XPath functions replies information about the status of the last call.

Function Declaration:

```
io:getUnitOfWorkID()
```

Returns the last created or received UOW as string.

Function Declaration:

```
io:isOK()
```

Returns true, if last call was successful. Otherwise returns false on error.

Function Declaration:

```
io:getContextData()
```

Returns the user context data, if UOW is received.

Function Declaration:

```
io:getException()
```

If an error occurs (`not( io:isOK())`), returns the error XML document. The main tag starts with `exception`.

UOW Transaction Handling

After sending or receiving a UOW, the client (=sender) or server (=receiver) must commit or back out (=rollback) the started transaction.

Element Extension:

```
<io:commitUOW/>
```

Commits the UOW. The Broker can deliver the created UOW to the partner or commit the correct processing pipeline step.

Element Extension:

```
<io:backoutUOW/>
```

Rolls back the UOW. The Broker does not deliver the created UOW to the partner, or delivery of the receiving UOW should be restarted later.

Logging

Sends log information to **System Log**.

Element Extension Example:

```
<io:log>
   <xsl:copy:of select="$result"/>
</io:log>
```

This example writes the contents of `$result` to the **System Log**.

Attributes:

Name	Value	Description
message	String	Writes this string.
select	XPath expression should return a string.	Writes this string.
encoding	XPath expression should return a string.	Defines the encoding of writing data.

Sending Mail

Sends a e-mail with the extension contents. Ask with `io:isOK()` for correct delivery.

Note: The property `mail.smtp.host` must be set correctly. The **JVM Property** page helps with parameter settings.

Caution: This function should be used only for control of your environment by sending mails to the administrator, for example. There is no support of document appendix setting.

Element Extension Example:

```
<io:sendMail
   subject="concat( //configuration[ @name = 'MainMenu' ]/title, ' sends hello ↩
world' )"
 to="//MAIL_TO_GW_ADMIN"
 from="//MAIL_FROM_GW_ADMIN">
     <xsl:copy:of select="$result"/>
</io:sendMail>
```

This example sends the contents of `$result` to the e-mail receiver.

Attributes:

Name	Value	Description
to	XPath expression should return a string.	E-mail receiver.
from	XPath expression should return a string.	Name of e-mail sender.
encoding	XPath expression should return a string.	Defines the encoding of writing data.
subject	XPath expression should return a string.	Title of e-mail.
content	XPath expression should return a string.	Sends this contents data.

Exchange Document

Sends and receives an XML document via HTTP-Post. It is possible to pass user ID and password to the web service with HTTP Basic Authentication.

Element Extension Example:

```
<xsl:variable name="resultFromWeb">
   <io:document href="'http://localhost:8080/axis/myWebService'">
        <xsl:copy:of select="$result"/>
   </io:document>
</xsl:variable>
```

This example sends the contents of `$result` to the HTTP server and stores the reply in `$resultFromWeb`.

Attributes:

Name	Value	Description
href	XPath expression should return a string.	HTTP address
userid	XPath expression should return a string.	HTTP Basic Authentication user ID.
password	XPath expression should return a string.	HTTP Basic Authentication user password.
select	XPath expression should return a string.	Send the string as text.
encoding	XPath expression should return a string.	Send the content with this encoding.
contentType	XPath expression should return a string.	Set this content type in the request header.

Name	Value	Description
contentType	XPath expression should return a string.	Set the `SOAPAction` request header to this value, e.g. `Some-URI`.

Getting System Parameter

The following function retrieves parameters from the System Constants page or parameters that have been passed via HTTP-Get request.

Function Declaration:

```
properties:getParameter( String key )
```

Returns the value as string of HTTP-Get parameter from key name. This HTTP-Get parameter should be sent to the caller in the URL or be passed as HTTP-Post parameter.

Function Declaration:

```
properties:getSystemConstancy( String key )
```

Returns the value as string of **System Constants parameter** from key name.

VI SAP R/3 Gateway Migration

31 SAP R/3 Gateway Migration

This part of the documentation provides background information on the migration of SAP R/3 Gateway to a new platform architecture within the webMethods Integration Server environment.

The product SAP R/3 Gateway (product code: SRG) will be retired on January 31, 2016. The reason for this is that SAP has announced the retirement of its RfcSDK interface which the product is using. This part of the documentation provides background information on the migration of SAP R/3 Gateway to a new architecture based on the webMethods Integration Server Environment. Please do not hesitate to contact Software AG for more details on the migration or a cost estimate for the migration project.

The project steps required to execute the migration by using the webMethods Integration Server SAP and the EntireX Adapter are described in detail; in the context of the documentation the product SAP R/3 Gateway will be referred to as *classic* SAP R/3 Gateway.

The migration assets (software, documentation and license) are combined under the product code SRGRP.

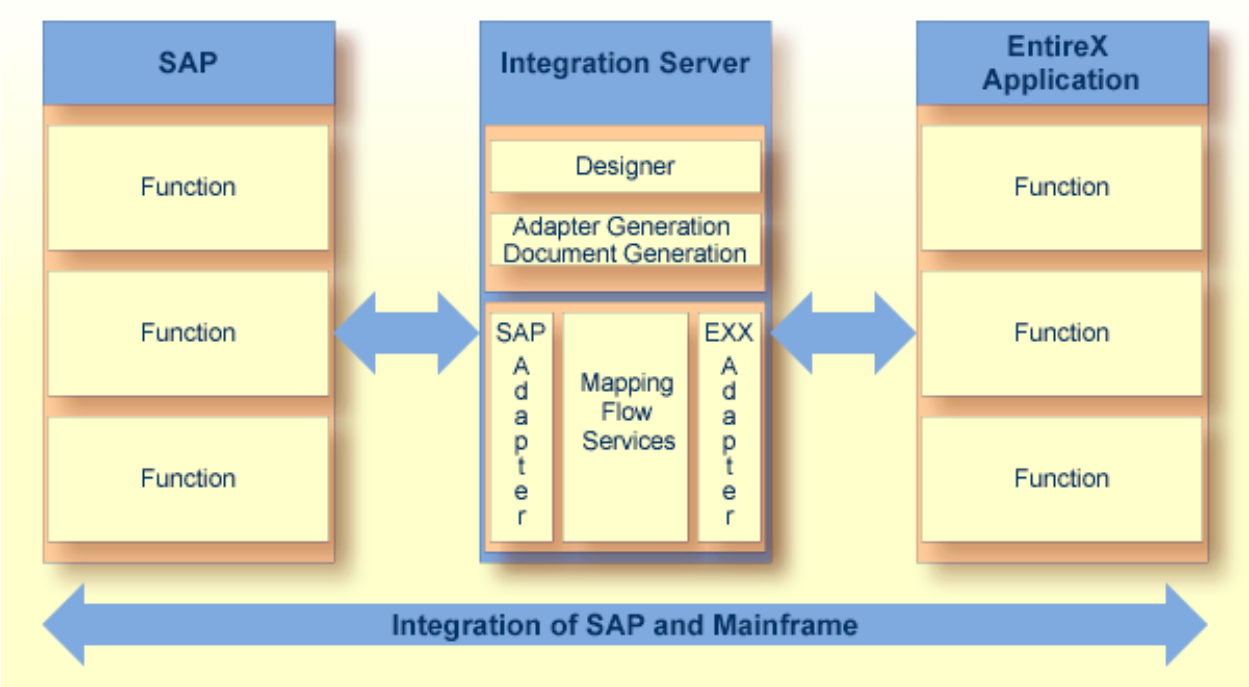
Motivation

The following points describe the motivation behind migrating the classic SAP R/3 Gateway to the new technology environment:

- SAP's deprecation of the classic RfcSDK interface used in SRG
- Availability of Unicode (previously not supported in SRG)
- Support of asynchronous communication (IDocs and tRfc)
- Extension of the limited point-to-point SRG architecture within the ESB
- Reduced support efforts for customers because of using Java instead of a C-based product
- Standard support of RFC data types such as `String`
- Auditing and logging support
- Use of caching features for distribution and big memory
- Cluster support for high availability (HA) to avoid single point of failure with active/active nodes

Architecture

The new architecture with its combination of webMethods Integration Server, EntireX Adapter and SAP Adapter based on Java replaces the classic SAP R/3 Gateway technology based on native C-programmed ERX and RFC interfaces.



The Designer is a prerequisite for developing the data transfer between the adapters.

Overview of the Migration Project and Prerequisites

The migration project lays great value on preserving all existing interfaces on the client or server side. No Rpc2Rpc Server clients (in most cases Natural RPC clients) or SAP clients (in the case of Rfc2Rpc) should require any changes. The existing middleware will be replaced with running Rpc2Rfc and Rfc2Rpc servers.

To achieve this goal, the IDLs of all implemented directions are used in the migration project. Ensure that the following prerequisites are met:

- Have the SRGRP license key of the webMethods Integration Server available.
- Set up the webMethods ESB environment with Integration Server, EntireX Adapter, SAP Adapter and Designer for development.

- Make IDLs available.
- (Re-) activate the test scenarios, know-how and environment of interfaces implemented with classic SAP R/3 Gateway. For example, if you have implemented unit tests, re-use these assets. If you have not used them in the past, plan for interface testing scenarios.
- Set up the webMethods ESB environment with Integration Server, EntireX Adapter, SAP Adapter for testing.
- The EntireX Adapter inside Integration Server requires RPC protocol version 2000 or higher.

  For **Migrating the Rpc2Rfc Server**, check your Natural (generated stubs) or EntireX RPC client environment to support this requirement.

  For **Migrating the Rfc2Rpc Server**, check your Natural or EntireX RPC Server environment to support this requirement.

  See the EntireX Release Notes for more about **Supported RPC protocols**.

About Data Mapping

Since the new architecture uses a different adapter technology, the following points must be taken into consideration for any existing or new testing scenarios when data is moved from one adapter to another.

There are usually two possibilities for the transport or mapping of data between interfaces. For functions or procedures, you can process the parameter data via calling conventions:

- Call by reference or
- call by value

When using `call by reference`, the sequence of the parameter types in the source must be identical to that in the target. When using `call by value`, the parameter name in the source must be identical to that in the target.

The classic SAP R/3 Gateway handled data processing in the following way:

- On data level 1, `call by value` was used because the importing, exporting or table name had to be identical.
- On data level 2, `call by reference`was used because the size of the data structure and types had to be identical.

In the Integration Server, the mapping of data is part of the pipeline handling. The implicit mapping works on `call by value` and the EntireX Adapter and SAP Adapter data is generated from IDL - the interface definition. In an ideal scenario all parameter names in IDL remain unchanged for

client and server and the implicit data mapping works perfectly. This means that no explicit data mapping needs to be defined.

If one or more parameter names differ, you have to apply the flow service data mapping with `call by value`.

If the mapping was thus changed from `call by reference` to `call by value`, tests must then be carried out to ensure that all data parameters have been passed from client to server.

Setting up the Development Environment

To set up the development environment

1. Install webMethods Integration Server with its license key.
2. Install Designer.
3. Import **WxSRG package** into Integration Server.
4. **Configure the Cache Manager**.

The new implementation in Integration Server will create a large number of assets such as connection, adapter, listener, document types and flow services, which have to be organized. The Integration Server supports the grouping of assets with packages in which the assets have a namespace. You can use the following statements to create packages and namespace.

To create packages and namespace

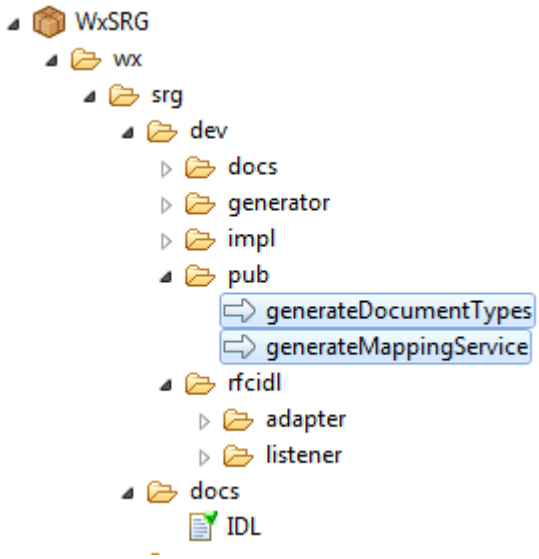
1. Create an Integration Server package for each IDL of Rpc2Rfc or Rfc2Rpc and give it the identical name in camel case. If you have multiple IDLs of the same type (e.g. Rpc2Rfc), add a postfix to identify the business case (for example `SD` for sales and distribution).
2. In a package, create a top folder and name it with the package name in lower case letters.
3. In the top folder, create the following subfolders:
   - a folder with the name `docs` for all document types;
   - a folder with the name `adapter` for all RFC or RPC functions and the connections;
   - a folder with the name `listener` for all RFC or RPC services acting as listener and for the connections.

WxSRG Package

The WxSRG package contains flow as well as java services for development and runtime:

- Services are available for generating Integration Server assets, e.g., document types during development.
- At runtime, services are available for handling functionalities of the classic SAP R/3 Gateway.

The development part is contained in the namespace `wx.srg.dev`.



There are 2 public services for generating Integration Server assets from IDL:

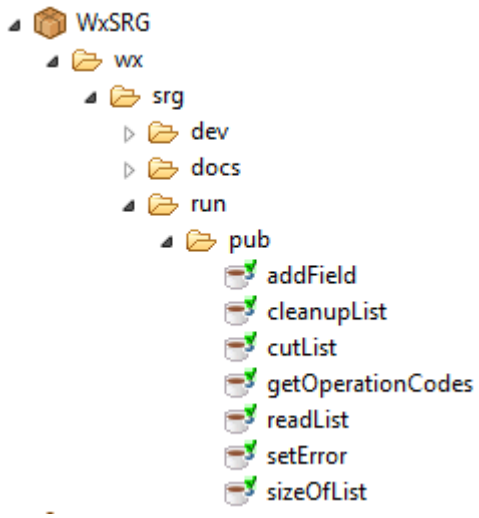
- `wx.srg.dev.pub:generateDocumentTypes`: Generates all required document types from IDL. One document type is generated for each group on level 1, another for all input parameters and another for the output parameters.
- `wx.srg.dev.pub:generateMappingService`: Generates one mapping flow service for each function in IDL.

Both services have input parameters:

Parameter	Description
filename	Sets the path and filename of the input IDL in the local filesystem or URL with HTTP Get protocol.
packageName	Sets the target package name of the generated output.
ns	Sets the namespace inside the package of generated output.

Parameter	Description
ns_of_docs	Sets the namespace of generated document types. This parameter is used on calling `wx.srg.dev.pub:generateMappingService` and the value comes from the `ns` parameter `off` calling `wx.srg.dev.pub:generateDocumentTypes`.

The services used during runtime are contained in the namespace `wx.srg.run`.



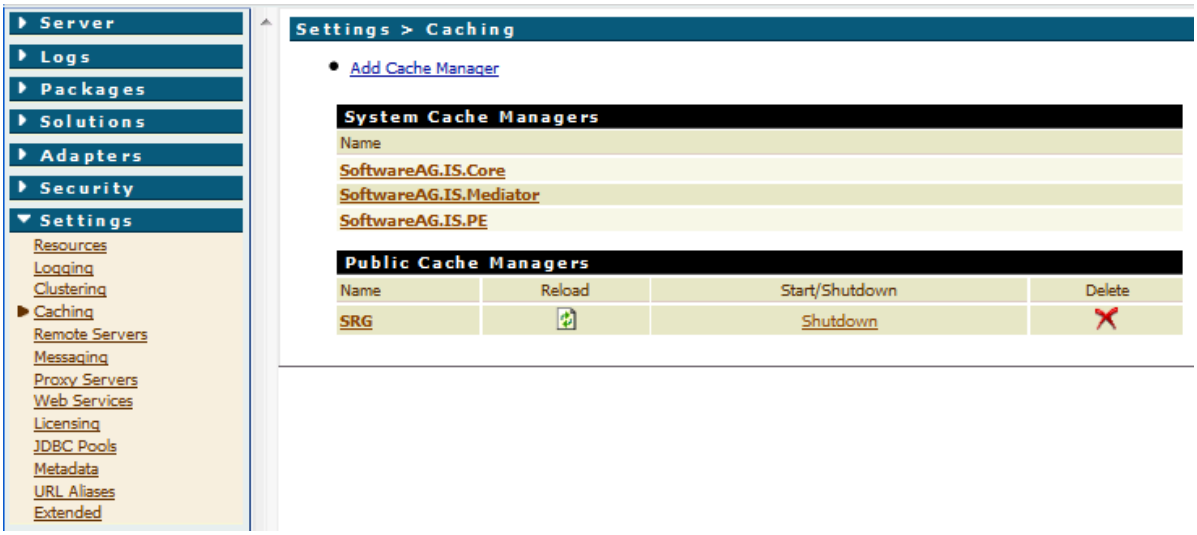
Note: Neither of the two services can overwrite flow-services or document types that have already been generated. If you want to regenerate the assets, you must delete them first.

Configuring the Cache Manager

The content for handling RFC tables is stored in the Integration Server cache. The SAP R/3 Gateway supports the cursor concept to read partial RFC tables; to set up this feature, you have to configure a Cache Manager named `SRG` in Integration Server.

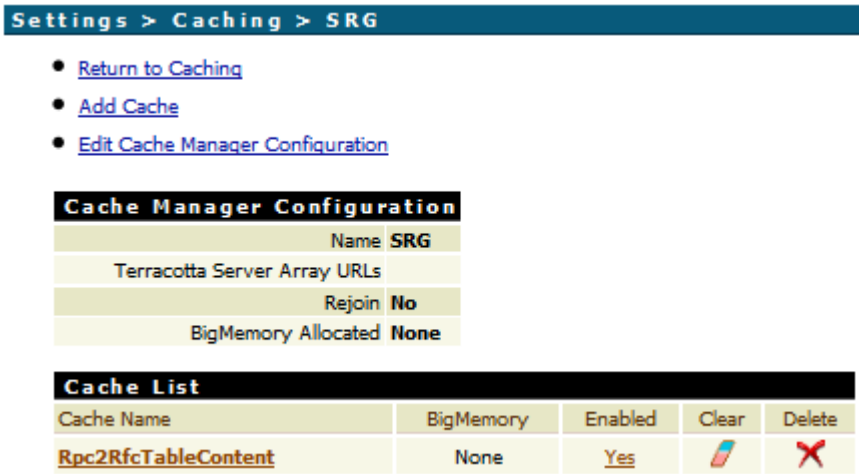
To configure the cache manager

1 In the Administrator, go to **Settings** > **Caching**.

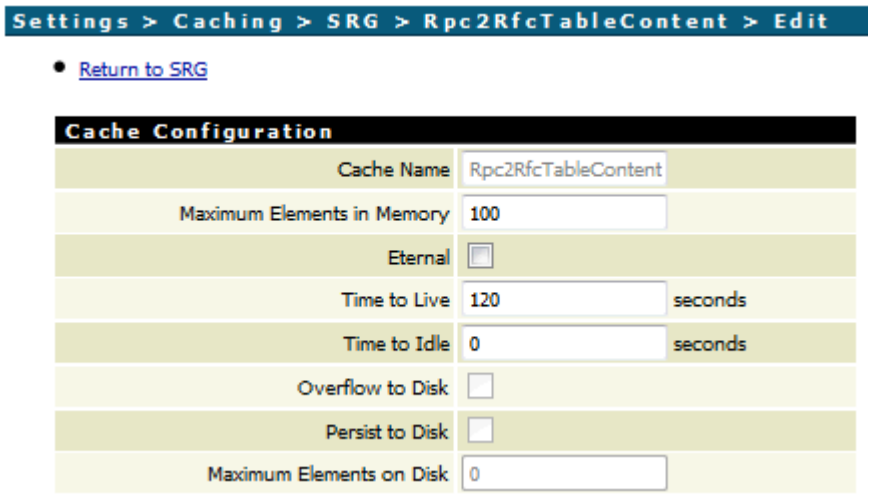


2 Inside `SRG` Cache Manager, add a cache named `Rpc2RfcTableContent`.

This results in



The cache contains the two required parameters `Maximum Elements in Memory` and `Time to Live`.



- `Maximum Elements in Memory`: Calculate the number of elements as the number of parallel sessions multiplied by the maximum number of tables in one RFC function. One element in the cache is one RFC table in one session. For example, if you have 10 parallel user sessions each calling an RFC function with 5 tables, you should set this value higher than 50.
- `Time to Live`: Set the no activity expiration time.

  Tip: You can set the same value of parameter **Connection Pool Time Limit** in Rpc2Rfc Server.

3 Set the other parameters to the default value or leave them blank for the Terracotta Server Array if you have no TSA.

Updating the RfcIdl Generator Tool

To support `Call by value`, the RfcIdl generator tool has a new option `webMethods SAP Adapter Calling Convention`, which allows printing the original SAP field names on level 2 into IDL.

To update the RfcIdl Generator Tool

1 Install the new option.

2 Upload the latest fix update into the `SAP R/3 Gateway` development environment.

3 Compile the RfcIdl tool: **Help** > **Setup Wizard** > **Compile EntireX RfcIdl Tool**.

The new option `webMethods SAP Adapter Calling Convention` is now available in **Development** > **Rpc2Rfc (SAP Server)**.

Note: The new option with value **yes** is now the default setting. To switch back to using camel case field names on level 2, you must select **no**.

Migrating the Rpc2Rfc Server

Each function inside the IDL must be implemented as a flow service, which is called from EntireX Adapter Listener for incoming requests. The incoming data is passed to the SAP Adapter function as a request document; the output is returned to the EntireX Adapter Listener as a response document.

Before using the generators to generate Integration Server assets, the IDL should be revised. The RfcIdl generator tool supports the exchange of parameter names on level 2. If the parameter names in the EntireX Adapter Listener and the SAP Adapter function are identical, the flow mapping service works with implicit data mapping. If the parameter names are not identical, explicit data mapping is required:

1. Use the **Compare** function to print a report about differences between the original parameter names in the SAP Repository and the IDL.
2. Use the **Replace** function to replace the current function in IDL with the interface definition from the SAP Repository.

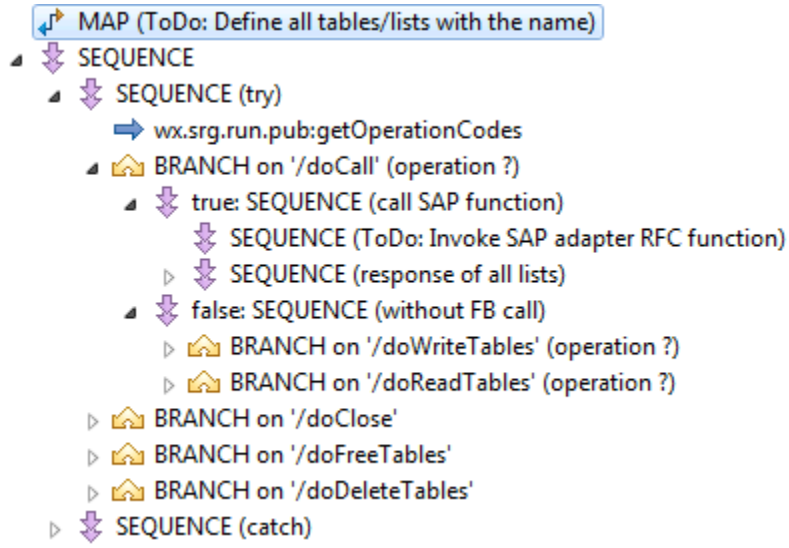
Notes:

1. The webMethods SAP Adapter uses the interface definition from the SAP Repository.
2. The webMethods EntireX Listener Notification uses the interface definition from IDL.
3. All differences between the IDL and the SAP Repository must be implemented in explicit data mapping services.
4. There are reserved names that cannot be used as field names in IDL. In most cases the reserved names are used on level 2 in SAP; you have to modify the IDL generated from the SAP repository before generating the EntireX Listener Notification. To do this, set the _ character as postfix in front of the following field names: `program`, `library`, `parameter`, `structure` and `version`.

For each IDL of the Rpc2Rfc server:

1. Generate the EntireX Listener.
2. Generate the SAP functions.
3. Generate the document types using `wx.srg.dev.pub:generateDocumentTypes` described in **WxSRG**.
4. Generate the mapping services using `wx.srg.dev.pub:generateMappingService` described in **WxSRG**.

One mapping services is generated for every function in the IDL; the mapping services are generated from template `wx.srg.dev.generator:templateMappingRpc2Rfc`; see the following figure:



For each function, the generated mapping services must be changed:

1. Check the EntireX Listener interface for input and output document types. The document types are generated with services `wx.srg.dev.pub:generateDocumentTypes` and have the suffix `Request` or `Response`. For **Input**, set document type `Request`, for **Output**, set document type `Response`. If the referenced document type is found, all input and output parameters are displaced. If not, check the parameter `ns_of_docs` on generation with `wx.srg.dev.pub:generateMappingService`.
2. Work through the generated mapping services and rework all steps that are commented with `ToDo`:
   - Enter the list of tables in the Map step `ToDo: Define all tables/lists with the name`. The table names of this SAP function must be set in the pipeline variable `DocumentNames` as String List.
   - In the Sequence step `ToDo: Invoke SAP adapter RFC function`, insert the SAP Adapter function generated from the SAP Repository.

Tip: See the flow services `wx.srg.dev.rfcidl.listener:RFCINTFB` or `wx.srg.dev.rfcidl.listener:RFCINTPA` in the **package WxSRG** for a sample implementation.

Using the helper service `wx.srg.run.pub:addField`

In case of different field names in the SAP Repository and the IDL on level 2, this helper service enables you to implement explicit mapping via `call by value`. The services adds a field with specific name and value into the document list.

The following example explains the use of the helper service:

1. The multiple group (RFC table) on level 2 has a field with the name `library`; however, the use of the parameter name `library` is not permitted because it is a reserved name in the IDL syntax and the field must be renamed to `_library` (for example).
2. Implement an explicit data mapping from field content `_library` to `library`; use the service by adding a field `library` in the target group.

Note: Removal of `_library` in the source is not required.

Migrating the Rfc2Rpc Server

Each function inside the IDL must be implemented as a flow service, which is called from the SAP Listener Notification for incoming requests. The incoming data is passed to the EntireX RPC function as a request document; the output is returned to the SAP Listener Notification as a response document.

1. For each IDL of the Rfc2Rpc server, generate the EntireX RPC adapter function.
2. For each function inside the IDL, generate a SAP listener notification.
3. One flow service exists for each generated SAP listener notification; inside this flow service, the generated EntireX RPC Adapter function is called.
4. All differences between the IDL and the SAP Repository must be implemented in explicit data mapping services.

Notes:

a. The interface of the flow services has the interface of the SAP Repository.

b. The interface of the EntireX RPC Adapter functions as the interface from the IDL.

Unsupported Features and Error Messages

The following table lists the unsupported features of classic SAP R/3 Gateway and the error messages in the IS package `WxSRG`.

Exception	Description
'write' only operation is not supported	The classic SAP R/3 Gateway supports the 'write' only operation (using operation code `OP_WRITE_TABLES` without `OP_CALL`) to transport data (content of RFC tables) to SAP without calling the RFC function in the first step. After the transport, the client can call the function in a second step with the operation code `OP_CALL`. This operation allows the sending of data (table items) from client to SAP server before the RFC function is called. This operation is not supported in the migration product. The preferred solution is to change the IDL to a variable array of this table.
No valid context handle created or handle is not available	If you call a query function in SAP that returns a large number of RFC table rows, the data are stored in the **cache** under a key (or called as context handle). To access the data, you must set the field `handle` with the last returned value. This error message is displayed if the value of handle is invalid.