webMethods

webMethods SAP Adapter

User's Guide

VERSION 6.5

webMethods, Inc.
South Tower
3877 Fairfax Ridge Road
Fairfax, VA 22030
USA
703.460.2500
http://www.webmethods.com

Document ID: ADAPTER-SAP-UG-65-20060217

Contents

About This Guide

This guide describes how to configure and develop applications for the webMethods SAP Adapter. It contains information for administrators who manage the system, and for application developers who create applications that use the system.

To use this guide effectively, you should:

- Understand the basic concepts of the SAP Adapter and XML.
- Have a general idea about how to perform basic tasks with the Developer.
- Know how to create flow services and/or Java services.
- Be familiar with the set up and operation of the webMethods Integration Server.

Related Documentation

The following documents are companions to this guide. Some documents are in PDF format and others are in HTML.

Refer to this book...	For...
webMethods Integration Server Administrator's Guide	Information about using the Server Administrator to configure, monitor, and control the webMethods Integration Server. This book is for server administrators. You will find this book at: *webMethods_directory*\IntegrationServer\doc\Integration ServerAdminGuide.pdf
webMethods Developer User's Guide	Information about creating and testing services and client applications. This book is for application developers. You will find this book at: *webMethods_directory*\Developer\doc\DeveloperUsers Guide.pdf
webMethods Developer Online Reference	Information about the controls in the Developer application windows and step-by-step procedures describing how to perform tasks with the Developer. You can access the online reference by clicking Help in an application window or dialog box.

Refer to this book...	For...
webMethods Built-In Services Reference Guide	This guide describes the built-in services provided with a standard installation of Integration Server and located in the WmPublic package. You will find this document at: *webMethods_directory*\Developer\doc\IntegrationServerBISReference.pdf
Serialization of ABAP data in XML	In depth information about how ABAP data is serialized in XML messages. This serialization is used for RFC and BAPI parameters. You will find this information at: http://ifr.sap.com/home/Documents/ABAP_Serialization.htm
XML Format Specifications	In depth information about how RFCs, BAPIs and IDocs are formatted in XML. You will find this guide at: http://ifr.sap.com/home/Documents/XML_index_E.htm
webMethods SAP Adapter IDoc Java API Documentation	In depth information about the webMethods SAP Adapter implementation of the "SAP Java IDoc Class Library". You will find this guide at: *webMethods_directory*\Integration Server\packages\WmSAP\pub\doc\api\index.html

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Italic	Identifies variable information that you must supply or change based on your specific situation or environment. Identifies terms the first time they are defined in text. Also identifies service input and output variables.
Narrow font	Identifies storage locations for services on the webMethods Integration Server using the convention *folder.subfolder:service*.
`Typewriter font`	Identifies characters and values that you must type exactly or messages that the system displays on the console.

Convention	Description
UPPERCASE	Identifies keyboard keys. Keys that you must press simultaneously are joined with the "+" symbol.
\	Directory paths use the "\" directory delimiter unless the subject is UNIX-specific.
[ ]	Optional keywords or values are enclosed in [ ]. Do not type the [ ] symbols in your own code.

SAP System Screens and Procedures

This guide contains some SAPGui images and procedures. Depending on the version of your SAP system and the platform on which it is run, your actual screens can appear slightly differently than the ones described in this guide. As a result, some SAP system procedures can also differ from what is described in this guide.

Additional Information

The webMethods Advantage Web site at http://advantage.webmethods.com provides you with important sources of information about the webMethods Integration Platform:

- Troubleshooting Information. webMethods provides troubleshooting information for many webMethods components in the webMethods Knowledge Base.
- Documentation Feedback. To provide documentation feedback to webMethods, go to the Documentation Feedback Form on the webMethods Bookshelf.
- Additional Documentation. All webMethods documentation is available on the webMethods Bookshelf.

webMethods

CHAPTER 1

About the webMethods SAP Adapter

What Is the webMethods SAP Adapter?

The webMethods SAP Adapter allows you to extend your SAP business processes and integrate non-SAP products using open and non-proprietary technology. The SAP Adapter allows for both asynchronous and bi-directional, real-time communication to and from the SAP system. You can:

- Execute SAP's implementation-independent Business Application Programming Interface (BAPI) methods, as they are described in the Business Object Repository (BOR). BAPIs are stable, precisely-defined, and well-documented interfaces to SAP solutions, providing standardized access to SAP solutions on a semantic level. You can quickly and easily create XML-based services that execute a BAPI. Applications within your organization can then invoke the services to execute a BAPI on the SAP system. Similarly, your business partners can make requests over the Internet to invoke a service that executes a BAPI. The BAPI-interfaces provide a unified access to the application-level functionality, independent of the type of call: Both synchronous and asynchronous processing can be triggered by using these interfaces.

  Asynchronous processing transparently uses the Application Link Enabling (ALE) services inside the SAP system for the client to integrate the business processes.

- Execute SAP Remote Function Calls (RFCs) from the SAP Adapter. You can access all SAP functionality that is available via RFC from the SAP Adapter. External applications do not need to understand SAP datatypes, ABAP structures or the RFC protocol to communicate with an SAP system.

- Send IDocs to the SAP Adapter. You can send Intermediate Documents to the SAP Adapter for further synchronous processing or let them be published to subscribers asynchronously.

- Call services from SAP systems. You can invoke services from an SAP system. This allows the SAP users to access information that is available via the SAP Adapter. The SAP Adapter enables integration between trading partners, thereby extending the reach of your SAP infrastructure to customers, partners, and suppliers.

- Route SAP business documents based on criteria you specify. The SAP Adapter provides rich routing capabilities for BAPIs, RFCs, and IDocs. There are different transport types available out-of-the-box. These include the routing of a business document to another SAP system or simply to a remote URL in an XML format.

The SAP Adapter allows you to increase customer loyalty and efficiency across the supply chain by tightly integrating your business infrastructure with that of any partner. Typical deployment scenarios for the SAP Adapter are:

- Real-time integration between supplier inventories and your SAP system
- Real-time integration between product, price, and availability information from any number of suppliers and your purchasing application
- Real-time integration between fulfillment and order tracking applications and your shippers' internal systems

Functional Highlights

- Synchronous and Asynchronous Communication with SAP systems via RFC and tRFC
- Bi-directional communication to and from SAP systems
- Higher level services to process SAP IDocs and BAPIs
- Easy XML and Internet enabling of existing SAP releases
- Support of BizTalk XML envelopes for BAPI and RFC calls
- Support of unified error handling of BAPIs and RFCs on XML level

Complete SAP System Integration

The SAP Adapter incorporates a full-fledged RFC Client and Server. These provide real-time bi-directional (outbound and inbound) communication to and from the SAP system. From an SAP application point of view, calling the SAP Adapter is no different from calling any other RFC server. The SAP proprietary RFC format is converted to XML so that no SAP software is needed on the other end of the communication line.

The SAP Adapter transparently supports both synchronous (RFC) and asynchronous (tRFC) calls from SAP systems. Thus, BAPIs and ALE scenarios are supported.

As a special service, SAP IDocs can be converted to a structured object with direct access to each single field. This allows you to modify an IDoc's contents on the fly. For example, if you want to customize incoming IDocs based on local data format, you can do so. You can also access existing BAPIs in SAP systems from a browser or client application, or you can send XML documents to the SAP system. Developing applications with the SAP Adapter requires no knowledge of SAP data structures, significantly reducing deployment time and cost.

Integrating SAP Systems Over the Internet

Most integration scenarios describe two systems communicating with each other securely over the Internet, despite existing firewall restrictions. The SAP Adapter leverages HTTP to seamlessly exchange data between two or more SAP systems across the Internet, without requiring changes to the existing security infrastructure.

In addition, the SAP Adapter transparently manages communication between different SAP system versions.

Routing IDocs, RFCs and BAPIs

The SAP Adapter provides rich routing capabilities for synchronous RFCs, asynchronous IDocs, and BAPIs, giving you better control of your trading relationships. Within minutes, you can configure the SAP Adapter to send an IDoc to another SAP system or to a remote URL in an XML format.

BAPIs provide a simple way to access SAP solutions. They can handle both synchronous and asynchronous calls to an SAP system using the same XML message format. Asynchronous BAPI calls are provided by using the ALE services inside the SAP system; you can leverage ALE mechanisms without having to deal with the sometimes complex IDoc format. This works for all IDocs that are generated from BAPIs.

Inside the SAP system, administrators can use the full bandwidth of services provided by ALE, including:

- Performance benefits through asynchronous processing
- Monitoring services
- Distribution services

BAPI interfaces are developed according to strict development guidelines, so they are:

- well-defined
- stable
- implementation-independent
- well-documented

The SAP Interface Repository provides public web-based access to the collection of BAPI interfaces provided by SAP and to the corresponding documentation.

In short, the use of BAPIs leads to the following advantages:

- SAP solutions are accessed on a standardized, implementation-independent level. Implementation on the SAP system can therefore be exchanged without invalidating the (BAPI) interface used by clients.
- A business management function which is implemented as a BAPI can be called both synchronously and asynchronously by using the same XML message.

Support for IDoc- and RFC-XML

The SAP Adapter is the de-facto XML interface to existing SAP systems releases. It supports all versions of IDoc- and RFC-XML (XRFC), as specified by the SAP-XML Specification. With the SAP Adapter SAP-XML interface, you can invoke RFCs via XML and convert IDocs to the SAP-XML format.

Note: The current version of RFC-XML is called *XRFC*.

Support of BizTalk XML Envelopes for BAPI and RFC Calls

By default, the SAP Adapter will use the standardized BizTalk XML envelope format for these XML messages. This simplifies data exchange with other Web messaging systems. The BizTalk XML envelope differentiates between an application-specific XML body and an XML header, which is used to exchange transport-specific information, for example for routing purposes. The BizTalk XML envelope can be used with both BAPI and RFC XML calls and also for both synchronous and asynchronous processing. See Appendix G, "Using BizTalk Envelopes with the SAP Adapter" for more information.

Support of Unified Error Handling of BAPIs and RFCs on the XML Level

To allow generic error evaluations on the client, regardless of the interface type (BAPI, RFC) used, the bXML format unifies the error handling of BAPIs and RFCs. In this way, interface type-specific error handling concepts (BAPI return parameter and function module exceptions) are converted on an XML level into a standardized type of error representation.

Built-in BAPI Tools

A set of tools to simplify the handling of BAPI calls has been integrated into the SAP Adapter.

For easy identification of the relevant application interfaces, the SAP Adapter includes a built-in BAPI Browser. This browser provides access to the interface data for each business object and its BAPIs and also serves as a search tool to the SAP interface world. With the built-in BAPI browser, you can quickly identify the necessary information to set up a routing notification for BAPIs.

Services to simplify the handling of BAPI transaction control have also been added to the SAP Adapter.

Transaction Store

The SAP Adapter provides a persistent transaction store that the transaction manager uses to track all IDocs and all tRFC calls routed to and from SAP systems via the webMethods Integration Server.

Architecture and Components

Basic Concepts

To use the SAP Adapter successfully, you should understand the following terms and concepts:

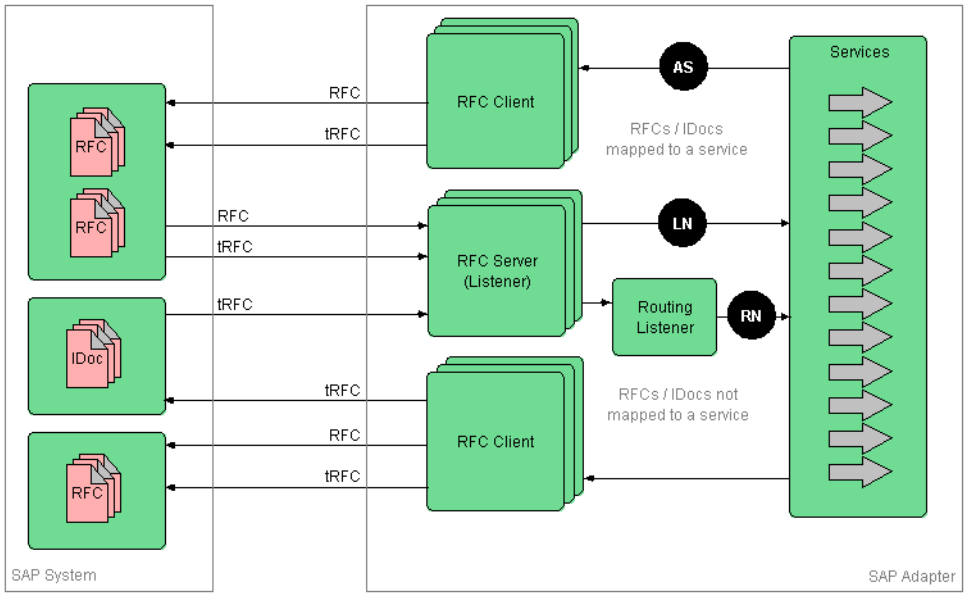
Term	Description
BAPI	Business functions in an SAP system that are written in the programming language ABAP. BAPIs are formalized RFCs. Systems remote from the SAP system commonly use BAPIs to have the SAP system perform an action.
RFC Server (Listener)	SAP terminology for a process that can accept Remote Function Calls (RFCs) from SAP systems. This allows SAP systems to access functions on external systems. In SAP Adapter terminology, this is a listener. Listeners are one or more threads on the SAP Adapter that wait for incoming requests from SAP systems. Listeners are named and registered with an SAP gateway to indicate that they are ready to accept requests. Listeners can accept RFC or tRFC requests.
RFC Client	SAP terminology for a process that sends RFCs to an SAP system to invoke functions.

Term	Description
tRFC (Transactional RFC)	Protocol for ensuring that an RFC is successfully executed and executed exactly once on the target system. The SAP Adapter can handle both inbound and outbound tRFCs.
tRFC protocol	Communications method that: ■ An SAP system uses to asynchronously invoke a function on a remote system. ■ A remote system uses to asynchronously invoke a function on the SAP system. The transactional RFC (tRFC) protocol ensures that an RFC is successfully executed and that it is executed exactly once.
TID	Transaction ID. A globally unique identifier used by tRFC to ensure exactly one execution. Can be up to 24 alphanumeric characters in length.
RFC	Requests that the SAP system initiates to have functions performed on remote systems, or calls remote systems initiate to have the SAP system perform a function.
RFC protocol	Communications method that the SAP system uses to synchronously invoke a function on a remote system, and a remote system uses to synchronously invoke a function on the SAP system.
Service	Integration Server functions that are named with a hierarchical folder/service syntax. Services can be flow, Java, or C/C++ developed by you, third-party vendors, or webMethods.
LIBRFC	Code library provided by SAP allowing third parties to integrate the RFC protocol into applications. This is how the SAP Adapter communicates with the SAP system.
IDoc (Intermediate Document)	EDI-like SAP business document.

Term	Description
Routing Listener	A process that accepts messages and routes them to a configured location, for example, messages can be routed to an SAP system, another Integration Server, or a remote URL in an XML format. **Note:** If you need to use a routing listener, you do not need to create one. The SAP Adapter includes a pre-configured routing listener called wm.sap.internal.ls:routingListener. For more information about the routing listener, see "Overview" on page 127.
Transaction Store	A repository that the SAP Adapter uses to track all transactions that pass through the SAP Adapter.

The following illustrates the components in a system that uses the SAP Adapter:

Architecture and Components

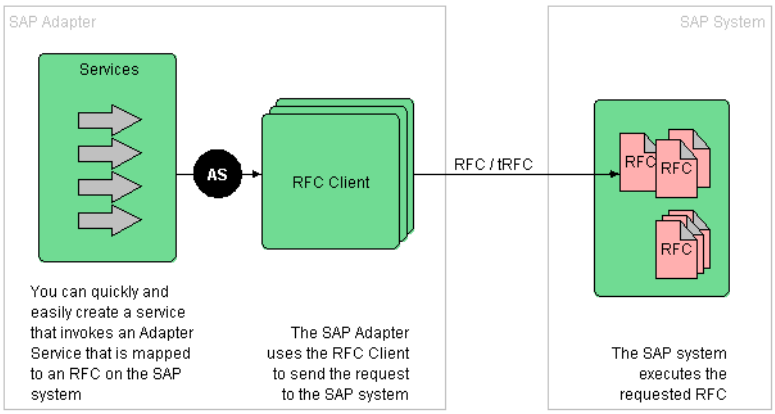


Invoking Business Logic

The SAP Adapter incorporates an RFC Server and RFC Client to provide real-time inbound and outbound communication to and from the SAP system.

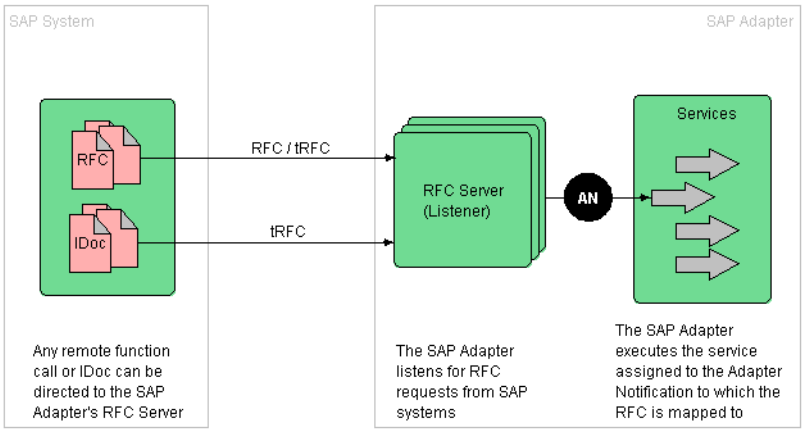
The SAP Adapter uses the RFC Client to send requests to execute RFCs on the SAP system.

Outbound Requests mapped to an Adapter Service



The SAP Adapter uses the RFC Server (Listener) to listen for incoming requests to execute services on the Integration Server. From an SAP system point of view, calling the SAP Adapter is no different from calling any other RFC server.

Inbound Requests mapped to an Adapter Notification

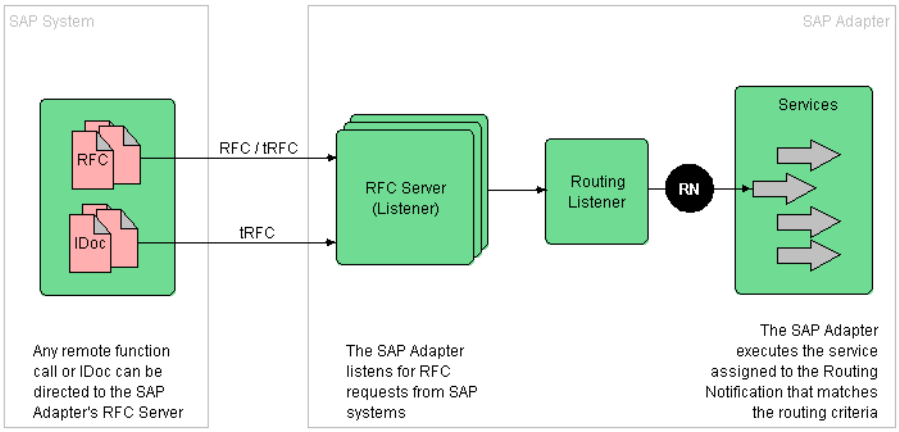


Sending Messages Through the Routing Listener

When the SAP Adapter receives a request on the RFC Server (Listener) that is not associated with a specific adapter notification, the SAP Adapter sends the request to the routing listener. The routing listener can receive:

- IDoc calls from an SAP system or IDoc-XML messages from any web client
- RFCs from an SAP system or RFC-XML and bXML messages from any web client

Inbound Requests into the Routing Listener

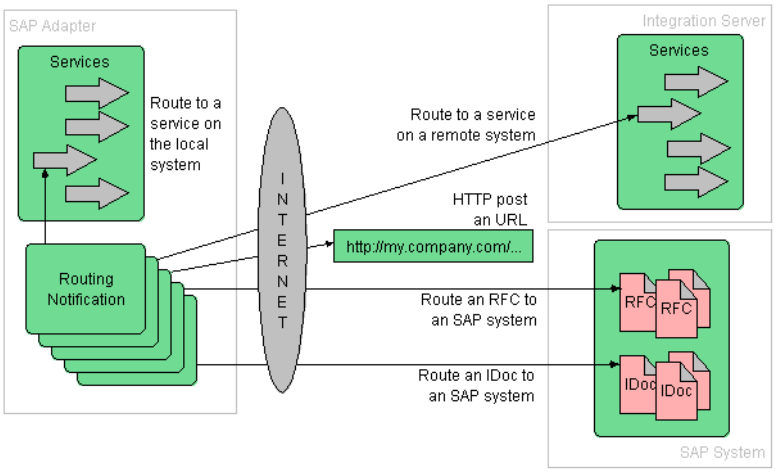


It then uses routing notifications to determine where to route the messages. The routing notification indicates where to route the message based on who the message is from (sender), who is to receive the message (receiver), and the message type.

A message can be:

- Routed to an SAP system (IDocs, RFCs and BAPIs)
- Routed to a service on the local machine or a remote machine
- HTTP posted to an URL

Outbound Transports Called from a Routing Notification



Note: An SAP Adapter inbound call is an outbound call from the SAP system's point of view.

Adapter Package Management

The SAP Adapter is provided as a package called WmSAP that you manage like any package on the Integration Server.

There are several considerations regarding how you set up and effectively manage your packages on the Integration Server, such as those described in the following list.

- Configure user-defined packages for your adapter connections and adapter services. See "Managing the Adapter Package" on page 42 for details.
- Understand how package dependencies work so you make the best decisions regarding how you manage your adapter services. See "Package Dependency Requirements and Guidelines" on page 43 for details.
- Control which development groups have access to which adapter services. See "Controlling Group Access" on page 46 for details.
- Understand how clustering, an advanced feature of the webMethods Integration Server, works to effectively manage your adapter services. See "Using the Adapter in a Clustered Environment" on page 47 for details.
- Enable and disable packages. See "Enabling and Disabling Packages" on page 44 for details.
- Load, reload, and unload packages. See "Loading, Reloading, and Unloading Packages" on page 45.

Adapter Connections

An adapter connection enables an SAP Adapter service to connect to an SAP system.

The adapter supports an RFC type of adapter connection. For instructions for configuring, viewing, editing, enabling, and disabling SAP Adapter connections, see Chapter 4, "Adapter Connections". For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide*.

For a list of tasks that you must do before you can create your connections, see "Before Configuring or Managing Adapter Connections" on page 56.

Connection Pools

The Integration Server includes a connection management service that dynamically manages connections and connection pools based on configuration settings that you specify for the connection. By default, connection pooling is enabled for all adapter connections.

A connection pool is a collection of connections with the same set of attributes. The Integration Server maintains connection pools in memory. Connection pools improve performance by enabling adapter services to reuse open connections instead of opening new connections.

Run-Time Behavior of Connection Pools

When you enable a connection, the Integration Server initializes the connection pool, creating the number of connection instances you specified in the connection's Minimum Pool Size parameter. Whenever an adapter service needs a connection, the Integration Server provides a connection from the pool. If no connections are available in the pool, and the maximum pool size has not been reached, the server creates one or more new connections (according to the number specified in Pool Increment Size) and adds them to the connection pool. If the pool is full (as specified in Maximum Pool Size), the requesting service will wait for the Integration Server to obtain a connection, up to the length of time specified in the Block Timeout parameter, until a connection becomes available. Periodically, the Integration Server inspects the pool and removes inactive connections that have exceeded the expiration period that you specified in Expire Timeout.

If the connection pool initialization fails because of a network connection failure or some other type of exception, you can enable the system to retry the initialization any number of times, at specified intervals.

For information about configuring connections, see Chapter 4, "Adapter Connections" on page 55.

Built-In Services For Connections

Integration Server 6.5 provides built-in services that enable you to programmatically control connections. You can use them to enable and disable a connection, and to return usage statistics and the current state (Enabled or Disabled) and error status for a connection. These services are located in the WmART package, in the pub.art.connection folder.

The built-in service pub.art.services:setAdapterServiceNodeConnection enables you to change the connection associated with an adapter service. See "Changing the Connection Associated with an Adapter Service at Design Time" on page 29.

For details, see the *webMethods Integration Server Built-In Services Reference*, which is available from the Help menu of the webMethods Developer.

Changing the Connection Associated with an Adapter Service at Design Time

The Integration Server provides a built-in service that you can use at design time to change the connection associated with an adapter service. This built-in service is named pub.art.service:setAdapterServiceNodeConnection. Using this service, you can change the specific connection associated with an adapter service at design time so that you do not need to recreate adapter services.

Note: This built-in service can be run at design time only; do not use it within an Integration Server flow or Java service. You must run this service directly from the Developer by selecting the service in the Developer and running it.

For details, see the *webMethods Integration Server Built-In Services Reference*, which is available from the Help menu of the webMethods Developer.

Other built-in services enable you to control connections; for more information, see "Built-In Services For Connections" on page 29.

Changing the Connection Associated with an Adapter Service at Run Time

Beginning with version 6.5, the Integration Server enables you to dynamically select the connection a service uses to interact with the adapter's resource. This feature enables one service to interact with multiple, similar backend resources.

For example, you can configure an adapter service to use a default connection that interacts with your company's production application server. However, at runtime you can override the default connection and instead use another connection to interact with the company's test application server.

For more information about overriding a service's default connection at runtime, see "Dynamically Changing a Service's Connection at Run Time" on page 62.

Adapter Services

Adapter services allow you to connect to the adapter's resource (that is, an SAP system) and initiate an operation on the resource from the Integration Server. You call adapter services from a flow or Java service to interact with function modules on an SAP system.

At design time, the adapter obtains information about the SAP document on the SAP system. You configure adapter services using the templates provided with the SAP Adapter. Each template represents a specific technique for doing work on a resource, such as invoking a function module on an SAP system. An adapter service template contains all the code necessary for interacting with the resource but without the data specifications. You provide these specifications when you configure a new adapter service.

You use the webMethods Developer to configure the adapter service. Some familiarity with using the webMethods Developer is required. See the *webMethods Developer User's Guide* for more information.

The SAP Adapter provides the following adapter service template:

Adapter Service Template	Description
RFC Adapter Service (synchronous)	Invokes an RFC on the SAP system, executes a tRFC, or confirms a transaction.

Adapter Listeners and Listener Notifications

The adapter uses an RFC listener to enable the SAP system to make Remote Function Calls to the Integration Server. It uses four different types of notifications to handle the RFCs passed through the RFC listener.

Listeners

A listener continually monitors an SAP system for RFC requests to send to the Integration Server. When a call is made from the SAP system, the listener passes the message to a listener notification.

A listener is a real-time process that you configure, enable, and disable using the Administrator. For detailed instructions on how to configure listeners, see "Listeners" on page 87.

Listener Notifications

A listener notification works in conjunction with a listener to filter and process the RFC requests in the SAP Adapter. The SAP Adapter uses four different listener notifications to process requests:

Listener Notification	Description
ALE Listener Notification (synchronous)	Handles incoming IDocs synchronously.
ALE Listener Notification (asynchronous)	Handles incoming IDocs asynchronously.
RFC Listener Notification (synchronous)	Handles incoming RFCs and tRFCs synchronously.
Routing Notification (synchronous)	Is used by the adapter to route all incoming requests that are not mapped to a service. The routing notification is a special case listener notification. See Chapter 8, "Routing Messages Through the SAP Adapter" for more information about routing notifications.

webMethods Manager Support for the Adapter

The webMethods Manager Server is a systems management facility based on the Open Management Interface (OMI). It automatically manages the SAP Adapter components if the Manager Server manages the Integration Server on which the SAP Adapter is installed. The managed SAP Adapter components are:

- Adapter connections
- Adapter services
- Listeners
- Listener notifications

See the *webMethods Manager Server Programmer's Guide* for information about the object interfaces available to the SAP Adapter. These object interfaces are the specific attributes and operations that you can perform to manage the SAP Adapter components.

Viewing the Adapter's Update Level

When using the SAP Adapter with Integration Server 6.5, you can view the list of updates that have been applied to the adapter. The list of updates appears in the Updates field on the adapter's About page in the Integration Server Administrator.

webMethods

CHAPTER 2

System Requirements

Overview

For detailed information on system requirements, see the *webMethods SAP Adapter Installation Guide*.

The webMethods SAP Adapter version 6.5 is supported on all platforms where both the webMethods Integration Server version 6.5 and SAP Java Connector are supported. For information on supported platforms for SAP Java Connector, see OSS note 0000549268.

CPU

For just "Internet-enabling" an SAP system, an average PC will be sufficient. For heavy-load scenarios, you should adjust your hardware accordingly.

Disk Space

Questions on Sizing

Appropriate sizing for the SAP Adapter installation strongly depends on the business scenario you want to establish. For specifying your individual requirements, you should answer the following questions first:

1. Is there a need for your own development, and if so, how big do you expect your scenario to be? Small Package: <1MB, average Package: <5MB, big Package: 10MB
2. What kind of data flow do you expect? That is, what kind of transactional messages and their size, how many per day, for how long do you want to keep them in the SAP Adapter transaction store after they are completed (`Confirmed`)
3. Do you need to keep statistical information of the kind written to audit.log and session.log? (See the *webMethods Integration Server Administrator's Guide* for an explanation of these two log files.)

Transaction Store

You must determine the storage space for the temporarily stored transactions. Use the following formula:

(Average size of a transactional message) x (Number of transactions per day) x (Number of days you want to maintain `Confirmed` transactions for reference).

To determine the average size of one transactional message

1 Determine the current size of the *webMethods_directory*/IntegrationServer/packages/WmSAP/txStore/ directory, including its subdirectories.

2 Send 100 typical transactional messages (transactions you will use in production) through the SAP Adapter.

3 Check the size of directory again.

4 Divide the difference by 100 to determine the average size of your transactional message. For example, a typical ORDERS IDoc is between 13kB and 35kB.

Note: You might want to schedule the pub.sap.transaction:sweep service to administer the growth of the transaction store. For more information about the service, see "pub.sap.transaction:sweep" on page 258.

Log Files

The log files need additional disk space. If the SAP Adapter runs on a high debug level during the implementation and test phase, it can write 1GB per day. Typical data volumes with a standard debug level (4 or lower) may generate between 10kB and 40MB per day (depending on traffic). The biggest parts are consumed by audit log and session log, which are used for statistical evaluations. You can turn off these logs to save disk space. Then the SAP Adapter needs only a few kB each day for the error log and server log. However, it is not recommended that you turn off these logs. This will only save a few kB and eliminates all possibilities for determining the cause of errors.

To turn off the logs

- Audit Log: Use the Edit Extended Settings feature of the Integration Server Administrator to view the `watt.server.auditLog` key and edit the key as follows: `watt.server.auditLog=off`.

Tip! You can also adjust this parameter by shutting down the Integration Server and adding the parameter to or editing it in the *webMethods_directory*/IntegrationServer/config/server.cnf file.

- Session Log: Use the Event Manager tool in the Developer to set the following events to the Enabled status "false":
  - Session End Event
  - Session Expire Event
  - Session Start Event
- Stats Log: Use the Event Manager to set Stat Event event to the Enabled "false".

Example of Minimum Disk Space

The following is an example of the minimum disk space for an average SAP Adapter:

Component	Component Required disk space (MB)
Basic Installation	80
Customer Packages (each)	~5
Transaction Store (1000 ORDERS per day, kept for two weeks)	280
Log files (kept for two weeks)	280
Total	645

Memory

Questions on Sizing

1 How many tasks will the Integration Server have to handle simultaneously? For example, how many RFCs, http requests, ftp connections, started listeners and RFC listeners will the Integration Server have to handle *simultaneously* at the peak time of the day? This determines the number of sessions *n*. For more information, see "Consumption Per Session" on page 37.

2 For how many SAP systems will the SAP Adapter register an RFC listener? How many threads do these listeners have?

3 How many messages (XML documents, IDocs, RFC calls) go through the Integration Server *simultaneously* at the peak time of the day? Is there any expensive mapping to be done?

Initial Consumption

The Integration Server needs approximately 80MB RAM to start up and just run without any heavy load on it. Depending on the load and expected traffic on the Integration Server, you need to add the following:

Consumption Per Session

Each task the Integration Server will handle (incoming http request, incoming/outgoing RFC call) requires one *session* on the Integration Server. If *n* is the number of sessions (see question 1), add the following amount of RAM:

½ *n* MB

If *n* proves to be much larger than 75, use the Edit Extended Settings feature of the Integration Server Administrator to view the `watt.server.threadPool` key and edit the key to enable the Integration Server to handle more than 75 tasks in parallel. The Integration Server then provides a pool of unused sessions, and the next request can be processed immediately without waiting for a new session to be created.

Note: If the size of your hardware does not support this level of RAM usage, do not change `watt.server.threadPool`.

You can also increase the value of the `watt.server.threadPoolMin=10` parameter to enhance performance.

Tip! You can also adjust these parameters by shutting down the Integration Server and adding the parameters to or editing them in the *webMethods_directory*/IntegrationServer/config/server.cnf file

For more information about these parameters, see the *webMethods Integration Server Administrator's Guide*.

Note: This procedure should be used with care to avoid unnecessary memory overload. For example, if you keep a pool of 1000 sessions and the Integration Server rarely needs more than 100, you will waste a lot of memory and resources.

Consumption for RFC Listeners

In the next step, you will need extra memory for every RFC listener thread (see question 2). For each listener thread, add 10MB. As this consumes a lot of memory, the number of threads for a listener must be chosen with care.

Note: If the number of threads is too small, the SAP system cannot send RFCs in parallel. The RFCs must be queued until an Integration Server thread is free to process them. The SAP system's work processes, trying to send RFCs, are then blocked and the SAP system's performance slows down considerably. On the other hand, if the number of threads is too high, the Integration Server consumes a lot of memory. This may slow down Integration Server performance. To optimize the number of threads, determine the average number of work processes the SAP system will use for sending RFCs.

Payload

You will have to calculate the amount of memory required to process documents simultaneously (see question 3). Use the following equation to calculate your needs:

Payload consumption = (document size x memFactor + RFCSize) x number of parallel documents, where:

memFactor = 3 + the number of INVOKE statements in the adapter or notification service (if there are mappings between document formats and different XML dialects) + the SSL value. The SSL value = 0 or 1, according to whether documents are encrypted with SSL or not.

RFCSize = 0, if no RFC is involved in transporting the document.

RFCSize = (1063bytes x number of segments + 524bytes) x number of IDocs in package, if the IDocs are either sent or received via RFC.

If you are in doubt about your assumptions for these values, you should run a few tests and watch by how much the memory actually increases during processing of one document.

Important! By default, the Integration Server starts with 128MB RAM. After you have calculated the memory that your SAP Adapter should use, you have to modify the setting:

```
set JAVA_MAX_MEM=128M
```

in *webMethods_directory*/IntegrationServer/bin/server.bat (or server.sh on UNIX) before starting the Integration Server.

Example of Memory Usage

The following is an example of a server handling 1000 ORDERS per day:

- The Integration Server has one standard RFC-listener with 3 threads.
- You expect a peak load of three documents in parallel, resulting in three parallel sessions. Allow another session for an Administrator or Developer to log on occasionally. Together with the session created by the RFC-listeners, this provides a total of eight parallel sessions and 4MB.
- The Integration Server will handle a peak of three documents in parallel, and the XML Package does some mapping when converting the IDoc to XML. For a relatively small document of 20kB in a 50 segment IDoc, this results in a memory consumption of (20kB x 6 + 52.4kB) x 3= 517.2kB (that can be rounded up to 1MB.)

Task Required	RAM (MB)
Integration Server core functionality	80
Sessions	4
RFC Listeners	30
Document processing	1
Total	115

In this example, you should be fine with approximately 128MB of memory for the Integration Server alone. After adding memory for the operating system and other tasks, a 256MB machine will be more than sufficient.

Miscellaneous

Encryption

When using SSL to encrypt XML documents, the time needed to process a document will increase by a factor of 3. In this case, you therefore need to take a CPU from the next higher class.

You also have to change the formula for calculating the RAM needed for processing documents: increase memFactor by one.

IDoc Packets

The performance can be vastly improved by using IDoc Packets of a well chosen size. If the packet size is to small, you will not take full advantage of the performance improvement. If it is to big, the Integration Server may run out of memory or start swapping. You should do some testing to find the optimum size; it depends on the IDoc type as well as on the size of your machine.

Logging

If the production scenario involves invoking thousands of services in a short period of time, you can achieve a huge performance improvement by turning off Audit Logging. For information about turning off logging, see "Log Files" on page 35.

webMethods

CHAPTER 3

Package Management

Overview

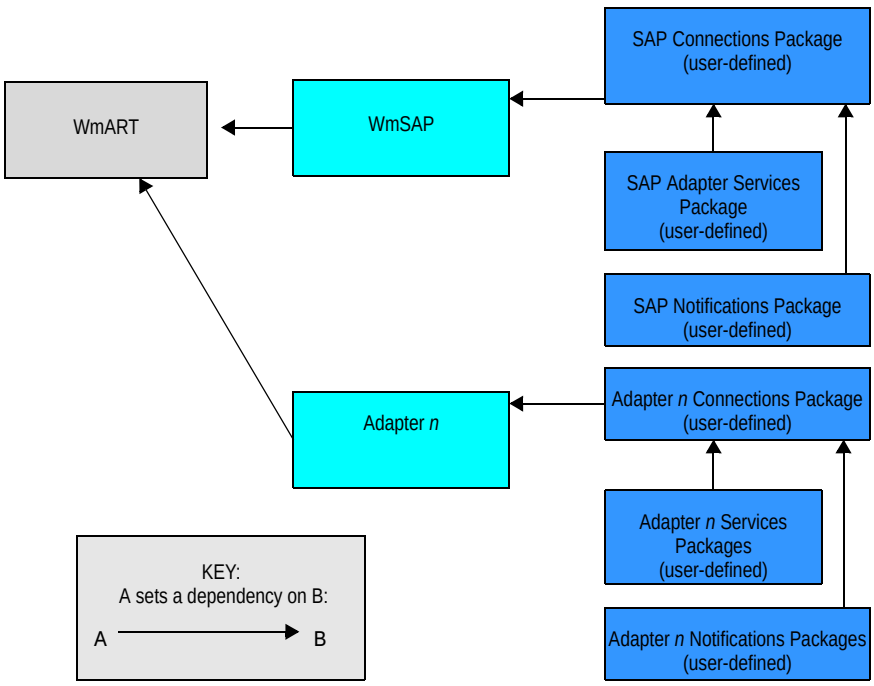
The following sections describe how to set up and manage your SAP Adapter packages, set up Access Control Lists (ACL), and use the adapter in a clustered environment.

Managing the Adapter Package

The SAP Adapter is provided as a package called WmSAP. You manage the WmSAP package as you would manage any package on the Integration Server.

When you create connections, adapter services, and adapter notifications, define them in user-defined packages rather than in the WmSAP package. Doing so will allow you to manage the package more easily.

As you create user-defined packages in which to store connections, adapter services, and adapter notifications, use the package management functionality provided in the Developer and set the user-defined packages to have a dependency on the WmSAP package. That way, when the WmSAP package loads or reloads, the user-defined packages load automatically. See the following diagram:



Package management tasks include:

- Setting package dependencies (see "Package Dependency Requirements and Guidelines" on page 43).
- "Enabling and Disabling Packages" on page 44.
- "Importing and Exporting Packages" on page 45.
- "Controlling Group Access" on page 46.

Package Dependency Requirements and Guidelines

This section contains a list of dependency requirements and guidelines for user-defined packages. For instructions for setting package dependencies, see the *webMethods Developer User's Guide*.

- A user-defined package must have a dependency on its associated adapter package, WmSAP. (The WmSAP package has a dependency on the WmART package.)
- Package dependencies ensure that at startup the Integration Server automatically loads or reloads all packages in the proper order: the WmART package first, the adapter package next, and the user-defined package(s) last. The WmART package is automatically installed when you install the Integration Server. You should not need to manually reload the WmART package.
- If the connections and adapter services of an adapter are defined in different packages, then:
  - A package that contains the connection(s) must have a dependency on the adapter package.
  - Packages that contain adapter services must have a dependency on their associated connection package.
- Keep connections for different adapters in separate packages so that you do not create interdependencies between adapters. If a package contains connections for two different adapters, and you reload one of the adapter packages, the connections for both adapters will reload automatically.
- The Integration Server will not allow you to enable a package if it has a dependency on another package that is disabled. That is, before you can enable your package, you must enable all packages on which your package depends. For information about enabling packages, see "Enabling and Disabling Packages" on page 44.

- The Integration Server *will* allow you to disable a package even if another package that is enabled has a dependency on it. Therefore, you must manually disable any user-defined packages that have a dependency on the adapter package before you disable the adapter package. For information about disabling packages, see "Enabling and Disabling Packages" on page 44.
- You can name adapter services and notifications the same name provided that they are in different folders.

Enabling and Disabling Packages

All packages are automatically enabled by default. When you want to temporarily prohibit access to the elements in a package, disable the package. When you disable a package, the server unloads all of its elements from memory. Disabling a package prevents the Integration Server from loading that package at startup.

To enable a package

1 Open the Integration Server Administrator if it is not already open.

2 In the Packages menu of the navigation area, click Management.

3 Click No in the Enabled column. The server displays a  and Yes in the Enabled column.

Note: Enabling an adapter package will *not* cause its associated user-defined package(s) to be reloaded. For information about reloading packages, see "Loading, Reloading, and Unloading Packages" on page 45.

Important! Before you manually enable a user-defined package, you must first enable its associated adapter package (WmSAP). Similarly, if your adapter has multiple user-defined packages, and you want to disable some of them, disable the adapter package first. Otherwise, errors will be issued when you try to access the remaining enabled user-defined packages.

To disable a package

1 Open the Integration Server Administrator if it is not already open.

2 In the Packages menu of the navigation area, click Management.

3 Click Yes in the Enabled column for the package that you want to disable. The server issues a prompt to verify that you want to disable the package. Click OK to enable the package. When the package is disabled, the server displays No in the Enabled column.

A disabled adapter will:

- Remain disabled until you explicitly enable it using the Administrator.
- Not be listed in the webMethods Developer.

Loading, Reloading, and Unloading Packages

If user-defined packages are properly configured with a dependency on the adapter package (as described in "Package Dependency Requirements and Guidelines" on page 43), at startup the Integration Server automatically loads or reloads all packages in the proper order: the WmART package first, the adapter package next, and the node package(s) last. You should not need to manually reload the WmART package.

Reloading Packages Manually

Reloading a user-defined package will *not* cause its associated adapter package to be reloaded. You can reload adapter packages and user-defined packages from either the Integration Server Administrator (by clicking the **Reload** icon on the Management window) or from the webMethods Developer (by right-clicking the package and selecting the **Reload Package** option from the menu).

Unloading Packages

At shutdown, the Integration Server unloads packages in the reverse order in which it loaded them: it unloads the node package(s) first, the adapter package next, and the WmART package last (assuming the dependencies are correct).

Importing and Exporting Packages

You import and export packages using the Developer. Exporting allows you to export the package to a .zip file and save it to your hard drive. The .zip file can then be imported for use by another package.

Important! Do not rename packages you export; the rename function is comparable to moving a package, and when you import the renamed package, you lose any triggers, connections, and notifications associated with this package

For details about importing and exporting packages, see the *webMethods Developer User's Guide*.

Setting Package Dependencies

You set package dependencies if a given package needs services in another package to load before it can load. For example, any packages you create for SAP Adapter services should identify the webMethods SAP Adapter package (WmSAP) as a package dependency because they require services in WmSAP to load first. Use the following guidelines:

- Set package dependencies from the adapter service package to the package containing the connection if you configure a connection in one package and the adapter services in another package. That is, the package that contains the connection should load before the adapter service package.

  When you set this package dependency, it ensures that if someone disables the connection package and then re-enables it, the adapter services will reload correctly.

- If both the connection and adapter services are in the same package, then no dependencies need to be set.

- In general, packages containing connections should have a dependency set to the adapter package itself. That is, the adapter service package should depend on the adapter connection package, which should depend on the adapter package. Similarly, if the adapter services are in the same package as the connections, the only dependency that you need to set is between the adapter connection package and the adapter package.

For more information about setting package dependencies, see the *webMethods Developer User's Guide*.

Controlling Group Access

To control which groups have access to which adapter services, use access control lists (ACLs). For example, you can use ACLs to prevent one development group from inadvertently updating the work of another group, or to allow or deny access to services that are restricted to one group but not to others.

For information about assigning and managing ACLs, see the *webMethods Developer User's Guide*.

Using the Adapter in a Clustered Environment

What Is webMethods Integration Server Clustering?

Clustering is an advanced feature of the webMethods platform that substantially extends the reliability, availability, and scalability of the webMethods Integration Server.

The clustering feature uses a shared *cluster store* to hold webMethods Integration Server state information and utilization metrics for use in load balancing and automatic failover support. Because this activity is transparent to the client, clustering makes multiple servers look and behave as one.

Clustering in general provides the following benefits:

- Load balancing. This feature, provided automatically when you set up a clustered environment, allows you to spread the workload over several servers, thus improving performance and scalability.
- Failover support. Clustering enables you to avoid a single point of failure. If a server cannot handle a request, or becomes unavailable, the request is automatically redirected to another server in the cluster.

Note: webMethods Integration Server clustering redirects HTTP and HTTPS requests, but does not redirect FTP or SMTP requests.

- Scalability. You can increase your capacity even further by adding new machines and webMethods Integration Server to the cluster.

For details on webMethods Integration Server clustering, see the *webMethods Integration Server Clustering Guide*.

Important! If you use clustering with the SAP Adapter, SAP transactions, like transactional tRFC calls and IDocs, will not be supported. You can only use non-transactional RFCs and BAPIs in a clustered environment.

About Clustering for the SAP Adapter

The evaluation of clustering has shown that its advantages in the specific SAP environment are restricted. As a result, webMethods generally recommends that you instead install several independent Integration Servers fronted by an external load balancer. The reasons for this recommendation are described in the following sections. Review the advantages and disadvantages of clustering in your specific environment carefully before deciding on the implementation.

In the light of these restrictions, clustering offers the following advantage and disadvantages.

Advantage:

- No additional load balancing component is needed.

Disadvantages:

- Load balancing for various kinds of HTTP clients is easier to achieve with an external load balancer and several instances of independent Integration Servers.
- Scalability through load balancing is not reached. Independent Integration Servers fronted by an external load balancer offer a better system performance.
- Failover mechanisms only makes sense for multi-step transactions. With an SAP backend system, these transactions are bound to a single Integration Server, so the checkpoint support of the cluster does not apply.

How Outbound RFCs Are Balanced

With regard to SAP logical unit of work (LUW) management, note the following additional restrictions.

Whenever it is necessary to maintain the user context in the SAP system, subsequent calls must be issued from the same Integration Server over the same RFC connection. This limits the use of clustering because an RFC connection cannot be put into the repository server. It is bound to a single Integration Server.

The method to bind an Integration Server session to an RFC connection is to embed subsequent requests of a SAP LUW (for example, a change BAPI and a BAPI_TRANSACTION_COMMIT) in a bind/release block. The resulting service can then be clustered.

However, if you want to execute an SAP LUW that has several dependent requests in a row that cannot be embedded into one service, you must ensure that the same user context is always used. You must also ensure that load balancing is prevented for each dependent call. You prevent load balancing (the redirection of your subsequent requests) by adding the requests to the *IntegrationServer_Directory*\config\redir.cnf file with the value “false”. For example:

```
<value name="pub.sap.client:lockSession">false</value>
```

You can start and finish your dependent request sequence by binding and releasing your session to your RFC connection using special services.

1 To bind your Integration Server session to an RFC connection, call the pub.sap.client:lockSession service before any series of requests that must have the same user context assigned at the SAP system host. For for more information about this service, see "pub.sap.client:lockSession" on page 225.

2 To release your Integration Server session, make sure that the LUW is complete (all services have been called). Then call the pub.sap.client:releaseSession service. For more information about this service, see "pub.sap.client:releaseSession" on page 226.

How Inbound RFCs Are Balanced

SAP listeners in the SAP Adapter implement self-registering RFC servers. Because they all have the same program ID, the SAP system gateway service automatically uses load balancing for any RFC calls to this program ID. Only one destination receives any single request; the gateway service ensures that as long as one of the listener threads is idle, then no requests from the SAP system will be blocked.

Configuring the Adapter in a Clustered Environment

When you configure the SAP Adapter to create adapter services, you must:

- Ensure that each Integration Server in the cluster contains an identical set of packages (see "Replicating Packages to webMethods Integration Servers" on page 49).
- Disable the redirection capability for certain predefined administrative services (see "Disabling the Redirection of Administrative Services" on page 50).

Replicating Packages to webMethods Integration Servers

Every webMethods Integration Server in the cluster should contain an identical set of packages that you define using the SAP Adapter; that is, you should replicate the SAP Adapter services and the connections, listeners, and listener notifications they use.

To ensure consistency, create all packages on one server, and replicate them to the other servers. If you allow different servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides in only one server, and that server is unavailable, the request cannot be successfully redirected to another server.

For information about replicating packages, see the chapter on managing packages in the *webMethods Integration Server Administrator's Guide*.

Disabling the Redirection of Administrative Services

As mentioned in "What Is webMethods Integration Server Clustering?" on page 47, a server that cannot handle a client's service request can automatically redirect the request to another server in the cluster. However, the SAP Adapter uses certain predefined administrative services that you should not allow to be redirected. These services are used internally when you configure the adapter. If you allow these services to be redirected, your configuration specifications might be saved on multiple servers, which is an undesirable result. For example, if you create two SAP Adapter services, one might be stored on one server, while the other one might be stored on another server. Remember that all adapter services must reside on all webMethods Integration Servers in the cluster.

To disable the redirection of administrative services

1. Shut down the Administrator. See the *webMethods Integration Server Administrator's Guide* for the procedure to do this.
2. Edit the following file:

   IntegrationServer_Directory\config\redir.cnf
3. Add the following lines to the file:

   ```
   <value name="wm.art">false</value>
   <value name="wm.sap">false</value>
   ```
4. Save the file and restart the webMethods Integration Server.

Clustering Considerations and Requirements

Note: The following sections assume that you have already configured the webMethods Integration Server cluster. For details about webMethods clustering, see the *webMethods Integration Server Clustering Guide*.

The following considerations and requirements apply to the SAP Adapter in a clustered environment.

Requirements for Each Integration Server in a Cluster

The following table describes the requirements of each Integration Server in a given cluster:

All Integration Servers in a given cluster must have identical...	For Example...
Integration Server versions	One Integration Server in the cluster cannot be version 6.5 and another Integration Server in the cluster be version 6.0.3–all servers must be version 6.5, with the same service packs and fixes (updates) applied.
Adapter packages	All adapter packages on one Integration Server should be replicated to all other Integration Servers in the cluster.
Adapter versions	In the cluster, all SAP Adapters must be the same version, with the same fixes (updates) and the same SAP libraries (versions) applied.

All Integration Servers in a given cluster must have identical...	For Example...
User-defined adapter packages	Each package that you have configured containing adapter connections, adapter services, flow services, listeners, triggers, and notifications for the SAP Adapter, must exist on all of the Integration Servers in the cluster, so that all Integration Servers in the cluster can handle any given request. The configuration and settings of all SAP Adapter connections, adapter services, listeners, triggers and notifications must be identical throughout the cluster. To ensure consistency, create all packages on one server, and replicate them to the other servers. If you allow different servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides in only one server, and that server is unavailable, the request cannot be successfully redirected to another server. For information about replicating packages, see the chapter on managing packages in the *webMethods Administrator's Guide*. Note: If you plan to use connection pools in a clustered environment, see "Considerations When Configuring Connections with Connection Pooling Enabled" on page 53.
Adapter connections	If you configure a connection to the application server, this connection must appear on all servers in the cluster so that any Integration Server in the cluster can handle a given request identically. If you plan to use connection pools in a clustered environment, see "Considerations When Configuring Connections with Connection Pooling Enabled" on page 53.
Adapter services	If you configure a specific adapter service, this same adapter service must appear on all servers in the cluster so that any Integration Server in the cluster can handle the request identically. If you allow different Integration Servers to contain different services, you might not derive the full benefits of clustering. For example, if a client requests a service that resides on only one server, and that server is unavailable, the request cannot be successfully redirected to another server.

See "Replicating Packages to webMethods Integration Servers" on page 49 for information about replicating adapter packages, connections, and adapter services across multiple Integration Servers in a cluster.

Considerations When Installing SAP Adapter Packages

For each Integration Server in the cluster, use the standard SAP Adapter installation procedures for each machine, as described in the *webMethods SAP Adapter Installation Guide*.

Considerations When Configuring Connections with Connection Pooling Enabled

When you configure an adapter connection that uses connection pools in a clustered environment, be sure that you do not exceed the total number of connections that can be opened simultaneously for that SAP system.

For example, if you have a cluster of two Integration Servers with a connection configured to an SAP system that supports a maximum of 100 connections opened simultaneously, the total number of connections possible at one time must not exceed 100. This means that you cannot configure a connection with an initial pool size of 100 and replicate the connection to both servers, because there could be possibly a total of 200 connections opened simultaneously to this SAP system.

In another example, consider a connection configured with an initial pool size of 10 and a maximum pool size of 100. If you replicate this connection across a cluster with two Integration Servers, it is possible for the connection pool size on both servers to exceed the maximum number of connections that can be open at one time.

For information about configuring connections for the SAP Adapter, see "Configuring Adapter Connections" on page 56.

For more general information about connection pools, see the *webMethods Integration Server Administrator's Guide.*

webMethods

CHAPTER 4

Adapter Connections

Overview

This chapter describes how to configure and manage SAP Adapter connections. For more information about how adapter connections work, see "Adapter Connections" on page 28.

Before Configuring or Managing Adapter Connections

To prepare to configure or manage adapter connections

1. Install the webMethods Integration Server and the SAP Adapter on the same machine. See the *webMethods SAP Adapter Installation Guide* for details.
2. Make sure you have webMethods administrator privileges so that you can access the SAP Adapter's administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.
3. Start your Integration Server and the Administrator, if they are not already running.
4. Using the Administrator, make sure the WmSAP package is enabled. See "Enabling and Disabling Packages" on page 44 for instructions.
5. Using the Developer, create a user-defined package to contain the connection, if you have not already done so. For more information about managing packages for the adapter, see "Managing the Adapter Package" on page 42.

Configuring Adapter Connections

Use the following procedure to define the parameters that the SAP Adapter will use to establish a connection to an SAP system. The SAP Adapter requires a connection to the SAP system whenever functionality from the SAP system is to be invoked; that is whenever the SAP Adapter acts as a client for an SAP system. But also when the SAP Adapter receives a call from an SAP system, it needs to make a call back to the calling system to look up the function interface or IDoc definition from the SAP Data Dictionary (DDIC).

To configure an adapter connection

1. In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.
2. On the Connections screen, click Configure New Connection.
3. On the Connection Type screen, click RFC Connection as the connection type.

4 For a basic connection, complete the following fields in the SAP Adapter section:

Field	Description/Action
Package	The package in which to create the connection. You must create the package using the Developer before you can specify it using this parameter. For general information about creating packages, see the *webMethods Developer User's Guide*. Note: Configure the connection in a user-defined package rather than in the adapter's package. See Chapter 3, "Package Management" for other important considerations when creating packages for the SAP Adapter.
Folder Name	The folder in which to create the connection.

5 In the Connection Properties section, complete the following fields using the values from your configured SAP system:

Field	Description/Action
Connection Alias	The RFC connection name. This is the name by which the SAP system will be known to SAP Adapter developers, clients, and partners.
User Name	The SAP user name.
Password	The SAP password.
Client	The three-digit SAP client number.
Language	The SAP language code. If connecting to a version 3 SAP system, this is one character. For a version 4 or later SAP system, it is two characters.

6 To configure logon properties, complete one of the following steps:

- To enable load balancing, configure the following fields:

Field	Description/Action
Load Balancing	If you select On, the group login concept is active. Complete the remaining fields in this step. If you select Off, you will connect to one dedicated application server. Skip to "Complete the following fields to set up the single server logon:" on page 58.
Logon Group	The name of the group you want to login.
Message Server	The host name or IP address of the message server.
System ID	The 3 digit ID of the SAP system.

- Complete the following fields to set up the single server logon:

Field	Description/Action
Application Server	The IP address or host name of the SAP system.
System Number	The SAP system number (00-99).

- Complete the following fields if you are using an external RFC Server:

Important! The settings should point either to the Gateway at which this RFC server is registered or to the Repository Server selected below. The load balancing and single server logon settings will be ignored if you configure an RFC server.

Field	Description/Action
External RFC Server	If you select Yes, an external RFC server will be used. Complete the rest of the fields in this section. If you select No, you do not need to complete the rest of the fields in this section.
Program ID	The Program ID of the external RFC Server
Gateway Host	Gateway Host for accessing your RFC Server. Must be the IP- or DN-address of the Gateway the RFC Server is registered on.

Field	Description/Action
Gateway Service	This corresponds to the system number of your RFC Server. Syntax: The Gateway Service must in the form of `sapgwXX`, where *XX* is a value from 00 to 99. Example: `sapgw07`
Repository Server	This server is taken from the existing SAP system list. All repository lookups such as metadata lookups for structures and function interfaces are done there. This helps using third-party RFC Servers from the SAP Adapter as client. They can provide their functionality without being extended by a specific interface, for example: RFC_GET_FUNCTION_INTERFACE

Note: Any server program that is enabled for RFC communication can be an external RFC Server. In order to use an external RFC Server with the SAP Adapter, the RFC Server must implement at least the following function modules:

— RFC_FUNCTION_SEARCH
— RFC_GET_FUNCTION_INTERFACE
— RFC_GET_STRUCTURE_DEFINITION (if the server is running with a 3.x Lib)
— DDIF_FIELDINFO_GET (if the server is running with a 4.x Lib)

The last three function modules are required only if the RFC Server is to be its own Repository Server.

7 Optionally, complete the following fields to enable security:

Field	Description/Action
SNC Enabled	Determines whether this server should use Secure Network Communications (SNC). Default: No.
SNC Quality of Service	SNC Quality of service, possible values: ■ Use global built-in default settings ■ Plain text, but authorization ■ Each data packet will be integrity protected ■ Each data packet will be privacy protected ■ Use maximum available security

Field	Description/Action
SNC Name	Your own SNC name if you do not want to use the default SNC name. This is the name you chose when generating a Personal Security Environment (PSE).
SNC Partnername	SNC name of the SNC partner (RFC server) or SNC name of the message server (Load Balancing).

8 Optionally, set the SAP Router String:

Field	Description/Action
SAP Router String	The SAP router string. A router string contains a substring for each SAP Router to set up a connection in the route: the host name, the port name, and the password, if one was given. Example: `/H/127.0.0.1/H/` Syntax: `/H/` indicates the host name. The SAP router string is only needed if there exists a firewall between the SAP system and Integration Server. For more information on configuring the SAP router, see the SAP Help Portal on the SAP Service Marketplace.

9 Optionally, complete the advanced settings for the connection:

Field	Description/Action
Use SAPGui	Run without/with/invisible SAP GUI between two RFC functions. Values: Off/On/Hidden Default: Off
RFC Trace	Disables/enables the creation of RFC trace information for this client connection. Values: Off/On Default: Off For detailed information on using the RFC trace, see Chapter 10, "Logging and Monitoring".
ABAP Debug	Run without/with the ABAP debugger Values: Off/On Default: Off

Field	Description/Action
Log transaction status	This switch can be set to `On` or `Off`. If set to `Off`, then the processing logs will not be saved, although a transaction will be created (or maintained), and the transaction can be monitored later on in the transaction list. **Tip!** Activating this switch reduces the amount of disk space needed and the time it takes to log the transaction status. It still allows you to check the current transaction status on the SAP Adapter.
Store message body	This switch can be set to `On` or `Off`. If set to `Off`, then the message body of the incoming document will not be stored to disk, although a transaction will be created (or maintained), and the transaction can be monitored later on in the transaction list. **Tip!** Activating this switch reduces the amount of disk space needed and the time it takes to persist the message body. It still allows you to track the message status on the SAP Adapter.

10 Click Save Connection to commit these changes.

Setting Up the SAP System for SNC Connections

If you want to use SNC connections for the communication, there are some more general and user specific settings required on the SAP system. These settings are listed below:

SAP system settings for SNC	Table/Parameter
General SAP system settings for SNC:	Table SNCSYSACL (SM30, view VSNCSYSACL, TYP=E) SNC name entry of the SAP Adapter.
Certificate log in: General Settings:	■ Profile Parameter: ■ SNC/libsapsecu=./libsecude.sl (for HP-UX) ■ SNC/extid_login_diag=1 ■ SNC/extid_login_rfc=1 ■ Table SNCSYSACL (SM30, view VSNCSYSACL, TYP=E), activate: ■ Certificate log in ■ Diag ■ RFC
Certificate log in: User specific settings:	Table USREXTID (view VUSREXTID), Category DN Entry with DN (Distinguished Name) from user certificate The Seq.No. 000 is the default entry. Set entry active.

For detailed information on the SAP application server settings for SNC, see your SAP documentation.

Dynamically Changing a Service's Connection at Run Time

For the SAP Adapter, you can change the service's connection in one of two ways.

The adapter service that ships with the SAP Adapter in the WmSAP package in the pub.sap namespace includes a field called *serverName*. Via this field, you can specify at run time the SAP system to be updated.

Additionally, Integration Server 6.5 enables you to override the default connection associated with the service at design time. To override the default, you must code your flow to pass a value through the pipeline into a service's *$connectionName* field.

For example, you have a flow whose primary purpose is to create an entry on the production SAP system. However, you want the flow to have the capability to create the entry on the test server, with the decision of which SAP system to update to be made programmatically at run time. The output signature of the flow's first service contains a field called *Target*. The flow could branch based on the value in *Target*.

- If *Target* contains the value "production," the second service in the flow would ignore *$connectionName*—thus using its default connection to connect to (and then update) the production SAP system.
- However, if *Target* contains the value "test," the second service in the flow would use the value in the *$connectionName* from the pipeline and connect to (and then update) the test SAP system.

Keep in mind that both connections—the default and override—must have the same SAP classes and methods.

For more information, see "Changing the Connection Associated with an Adapter Service at Run Time" on page 30.

Enabling Adapter Connections

A connection must be enabled before you can configure any adapter service using the connection, or before an adapter service can use the connection at run time. You enable adapter connections using the Administrator.

Note: When you reload a package that contains enabled connections, the connections will automatically be enabled when the package reloads. If the package contains connections that are disabled, they will remain disabled when the package reloads.

To enable a connection

1 In the Adapters menu in the Administrator's navigation area, click SAP Adapter.

2 On the Connections screen, click No in the Enabled column for the connection you want to enable.

The Integration Server Administrator enables the adapter connection and displays a  and Yes in the Enabled column.

- If you receive an error message, click the Edit button to view the configuration. Verify that your configuration information is correct. See "Editing Adapter Connections" on page 64.
- In case of an RFC error message, an RFC trace file named dev_rfc.trc will be written to the *webMethods_directory*\IntegrationServer directory. This file contains a more detailed description of the error cause.

Viewing Adapter Connection Parameters

You can view a connection's parameters from the Administrator or from the Developer.

To view the parameters for a connection using the Administrator

1 In the Adapters menu in the Administrator's navigation area, click SAP Adapter.

2 On the Connections screen, click the View icon  for the connection you want to see.

The View Connection screen displays the parameters for the connection. For descriptions of the connection parameters, see "Configuring Adapter Connections" on page 56.

3 Click Return to SAP Adapter Connections to return to the main connections screen.

To view the parameters for a connection using the Developer

1 Start the Developer if it is not already running.

2 From the Developer's navigation area, open the package and folder in which the connection is located.

3 Click the connection you want to view.

The parameters for the connection appear on the Connection Information tab. For descriptions of the connection parameters, see "Configuring Adapter Connections" on page 56.

Editing Adapter Connections

If a connection parameter changes, or if you want to redefine parameters that a connection uses when connecting to an SAP system, you can update a connection's parameters using the Administrator.

After you edit a connection, you must reload the WmSAP package for the change to take effect.

To edit a connection

1 In the Adapters menu in the Administrator's navigation area, click SAP Adapter.

2 Make sure that the connection is disabled before editing it. See "Disabling Adapter Connections" on page 66 for instructions.

3 On the Connections screen, click the **Edit** icon  for the connection you want to edit.

The Edit Connection screen displays the current parameters for the connection. Update the connection's parameters by typing or selecting the values you want to specify.

For descriptions of the connection parameters, see "Configuring Adapter Connections" on page 56.

4 Click **Save Changes** to save the connection and return to the Connections screen.

Copying Adapter Connections

You can copy an existing SAP Adapter connection to configure a new connection with the same or similar connection properties without having to re-type all of the properties for the connection. You copy adapter connections using the Administrator.

To copy a connection

1 In the **Adapters** menu in the Administrator's navigation area, click **SAP Adapter**.

2 On the Connections screen, click the **Copy** icon  for the connection you want to copy.

The Copy Connection screen displays the current parameters for the connection you want to copy. Name the new connection, specify a package name and folder name, and edit any connection parameters as needed by typing or selecting the values you want to specify.

Note: When you copy a connection, the new connection does not save the password of the original connection. You must enter and then retype the password before you can save the new connection.

For descriptions of the connection parameters, see "Configuring Adapter Connections" on page 56.

3 Click **Save Connection** to save the connection and return to the Connections screen.

Deleting Adapter Connections

If you no longer want to use a particular SAP Adapter connection, you can delete it by following the instructions in this section. You delete adapter connections using the Administrator.

If you delete a connection, the adapter services or notifications that are defined to use the connection will no longer work. However, you can change which connection an adapter service uses. Therefore, if you delete a connection, you can assign a different connection to an adapter service and re-use the service. To do this, you use the built-in webMethods function setAdapterServiceNodeConnection. For more information, see "Changing the Connection Associated with an Adapter Service at Design Time" on page 29.

To delete a connection

1 In the Adapters menu in the Administrator's navigation area, click SAP Adapter.

2 Make sure that the connection is disabled before deleting. To disable the connection, click Yes in the Enabled column and click OK to confirm. The Enabled column now shows No (disabled) for the connection.

3 On the Connections screen, click  for the connection you want to delete.

The Integration Server deletes the adapter connection.

Disabling Adapter Connections

SAP Adapter connections must be disabled before you can edit or delete them. You disable adapter connections using the Administrator.

To disable a connection

1 In the Adapters menu in the Administrator's navigation area, click SAP Adapter.

2 On the Connections screen, click Yes in the Enabled column for the connection you want to disable.

The adapter connection becomes disabled and you see a No in the Enabled column.

Testing the Execution of an RFC

After you have verified that you can connect to an SAP system, you can use the following procedure to verify that the SAP system accepts and processes an RFC request from the SAP Adapter.

To test an RFC from the SAP Adapter

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 In the SAP Adapter menu, click Lookup.

3 From the drop down list in the System ID field, select the system ID of the SAP system that hosts the function module you want to test.

4 In the Function Name field of the Function Search group, enter the name of the function module you want to test. For example, enter `RFC_FUNCTION_*`. Click Search. The SAP Adapter searches for all RFCs that begin with RFC_FUNCTION_.

5 In the list of matching RFCs that is returned, follow the link of the RFC you want to test. For this example, select RFC_FUNCTION_SEARCH. The SAP Adapter displays the function signature for the selected function module. It lists the imports, exports, and tables comprising this function module.

6 Select Test Function to invoke the function module on the SAP system you selected in step 3.

7 Enter `RFC_*` in the field FUNCNAME and click Test Function again to proceed. After a few moments, a screen displays the number of functions found.

8 Follow the entries link beside this row count to view the functions found.

Testing the Execution of an RFC-XML

You can invoke a function module with RFC-XML from the SAP Adapter user interface via the Lookup screen. This screen contains an RFC-XML template you can use to invoke function modules. Perform the following procedure to access the Lookup screen and invoke a function module via RFC-XML.

To invoke a function module via XML from the SAP Adapter

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 In the SAP Adapter menu, click Lookup.

3 Enter the function module you would like to invoke via XML in the Function Name field of the Function By Name section.

4 Select RFC-XML.

This generates a form containing the XML template necessary to invoke the function module. Fill in the appropriate inputs and select Invoke on System ID. You will see the XML response in the browser (you may have to select View Source to see the actual XML source in your browser).

Testing the Execution of a BAPI Via XML

The lookup-tool comes with a built-in BAPI Browser. You can use this tool for an overview of the BAPI interfaces inside your SAP system, and also to directly call a BAPI via XML.

Starting to Browse

You can start browsing your BAPIs using the Lookup tab. Here you can enter the name of a business object and a BAPI. This selection always applies to the SAP system selected in the field System ID.

There are several ways you can fill out the form:

- In the group BAPI By Name enter the full name of the business object and BAPI. You can either view the details of the BAPI by selecting Lookup or directly invoke the BAPI via XML by clicking bXML.
- In the group BAPI By Name enter only the name of the business object. Select Lookup to view details of the business object and select one BAPI from the list of available BAPIs for the business object.
- In the group BAPI By Name enter * in the business object field to view a list of all business objects available in the selected system.

Note: Business object names and BAPI names are case sensitive.

Displaying a List of Business Objects in the System

By entering * in the business object field on the Lookup main page, you can view a list of all business objects available on the selected system.

Each link in this list represents a business object. If you follow the link, you will get a detailed view of this business object. This list can also contain business objects that do not contain BAPI methods.

Displaying a Business Object

By entering a correct name of a business object in the corresponding field on the Lookup main page, you can view detailed information for this business object. The detailed information consists of a list of all the key fields of a business object and a list of all its BAPI methods. For business objects, which only provide instance-independent methods ("static" methods in programming languages like C or JAVA), the Key fields list is empty.

For business objects that do not provide BAPIs, the list of BAPIs will be empty.

Just follow the links to the corresponding table fields to display more information on a specific key field or BAPI.

Displaying a BAPI

By entering the full name of the business object and BAPI in the corresponding field on the Lookup main page, you can display detailed information of the BAPI. You can also display this information by following the corresponding link on the display page for a business object.

Field	Description
Static method	The Static method flag is true if the BAPI is an instance-independent method. In practice, that means that you do not have to specify the business objects key fields when calling this method.
Dialog method	The Dialog method flag is true if the method is a dialog method. If the method is not using a dialog, the flag is false. Dialog BAPIs require a SAPGui.
Factory method	The Factory method flag is true if the BAPI is used to create an object instance inside the SAP system. Factory methods return the key fields of the business object.
Implementing RFC	The Implementing RFC is the message type that has to be used to set up routing for synchronous calls coming from an SAP system. This is technically realized via RFC, so here the name of the corresponding RFC function module inside the SAP system is provided. Although it is possible at the moment, it is not recommended to call BAPIs directly via RFC-XML, as this would only be an implementation-dependent view of the BAPI.

Field	Description
ALE message type	The ALE message type is the message type that has to be used to set up routing for asynchronous calls coming from an SAP system. This is technically implemented via IDoc transfers, but when using the BAPI styled XML calls, the ALE message type is only needed to set up routing.
Parameters	The Parameters field contains a list of all parameters in the BAPI interface. By selecting this link, you can display more information on each parameter. Please note that the key fields of a business object are not displayed in this parameter area but in the Key fields area on the business object detail page.

By clicking the Create XML template button, you can generate the XML message that has to be used to invoke this BAPI via XML. After it has been generated, you can immediately send the XML to the SAP system for testing purposes (see below).

Displaying a BAPI Parameter

By selecting a specific parameter from the list of parameters provided on the BAPI detail page, you can display additional information for this parameter.

Field	Description
Internal Name	Parameter name of the underlying RFC.
Direction	The use-direction of the parameter is displayed. BAPIs use importing, exporting and changing (= importing and exporting) parameters.
Optional	If the flag Optional is set to true, you can omit this parameter in an XML message. Usually, the implementation inside the SAP system uses specific default values for optional parameters that were omitted.
Table	If the flag Table is set to true, this parameter may consist of several lines. Each line has the data type specified in the ABAP Dictionary Type field.
ABAP Dictionary Type	The ABAP Dictionary Type is a link to the data type description used for this parameter. This may be a whole structure or just a field of a structure. By following the hyperlink, you will always see the whole structure.

Displaying a Key Field

By selecting a specific key field from the list of key fields provided on the business object detail page, you can display additional information for this key field.

Field	Description
Internal Name	Parameter name of the underlying RFC.

ABAP Dictionary Type	The ABAP Dictionary Type is a link to the data type description used for this parameter. This is usually a field of a structure inside the SAP Data Dictionary. By following the hyperlink, you will see the whole structure.
	
	Note: Key fields are used with instance-dependent (non-static) methods and with factory methods.
	
	In calls to these methods, the key fields of the corresponding business object have to be specified as attributes of the business document root element.
	
	For example, using the key field `CompanyCodeId` in a call to the `CompanyCode.GetDetail`:
	
	`<?xml version="1.0" encoding="iso-8859-1"?>`
	`<biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">`
	`  <header>`
	`  …..`
	`  </header>`
	`  <body>`
	`  <doc:CompanyCode.GetDetail CompanyCodeId="0001"`
	`  xmlns:doc="urn:sap-com:document:sap:business"`
	`  xmlns="" >`
	`  </doc:CompanyCode.GetDetail>`
	`  </body>`
	`</biztalk_1>`

Generating XML Calls for a BAPI

By entering the full name of the business object and BAPI in the corresponding field on the Lookup main page and then choosing Create XML Template, you can generate an XML message representing the call to this BAPI. You can also generate this message by following the corresponding link on the display page for a BAPI.

The page displays the XML message needed to call the BAPI which is wrapped in the BizTalk XML envelope. You can edit the XML and enter your specific data into the pre-generated XML elements for parameters and key fields.

You can also adjust the pre-generated information in the BizTalk XML header fields that are filled with default data.

There are three ways of calling the BAPI, which can be selected through the list:

- Synchronously calling the BAPI in an SAP system.
- Asynchronously calling the BAPI in an SAP system. This only works for BAPIs that are mapped to an ALE interface. To see whether the BAPI is mapped to an ALE interface, check the BAPI detail page. If an ALE message type is specified, the BAPI can be called asynchronously.
- Applying routing notification: The call will be sent to the BAPI inbound process of the routing listener that will check if a routing notification for this BAPI call exists and will forward the call to the specified service. This will only work if a correct routing notification has been setup, otherwise you will receive an XML error message.

After clicking Invoke, the result of your XML call is displayed. In the case of synchronous processing, this will be a BizTalk message with either the exporting parameters of the BAPI or with an error message.

For asynchronous calls, this will be a BizTalk message with an empty body in the case of success or with an error message, if the message could not be delivered to an SAP system. Application specific errors are not returned in the case of asynchronous calls, but you can use the ALE monitoring tools in the target SAP system to check these errors.

Posting a BAPI from an HTTP client

For information about calling a BAPI from an external client using an XML document, see Chapter 10, “Routing BAPIs through the SAP Adapter”.

webMethods

CHAPTER 5

Adapter Services

Overview

Adapter services for the SAP Adapter work by executing a function module residing on an SAP system. A function module performs functions against data in the SAP system.

You can create a service for any function module that is designated for external use (for example: 'remote enabled'). This includes all BAPIs, which are formalized function modules.

Before you can create an adapter service, you must configure an RFC client connection to the SAP system you want to access. You identify a connection alias and specify information that is required to connect to the SAP system. To create an adapter service, you select the SAP system and the RFC that you want the service to execute. After creating the adapter service, you can create a client that invokes the service.

This chapter describes how to create an adapter service for a specific function module on one specific SAP system and how to test the adapter service.

Note: You must have administrator privileges on the Integration Server to create a new RFC connection to an SAP system. If you do not have such privileges, have your Integration Server administrator create the new RFC connection for you.

Creating an Adapter Service that Executes an RFC

To create an adapter service that executes an RFC, select the RFC Adapter Service template. This feature allows the function module to be exposed as a standard service. After you create the adapter service, business partners interested in calling this service can incorporate it into their client code.

Note: Calling a BAPI according to its definition in the BOR is currently not possible as an adapter service. In this case, use the BAPI transport.

Note: An SAP Adapter outbound call is an inbound call from the SAP system's point of view.

Terminology

To use the SAP Adapter successfully, you should understand the following terms and concepts:

Term	Description
Export	Output parameters of a function module. Output parameters are simple values or structures.
Function Signature	Specifications of the import, export, and table parameters of an function module. Also known as a *function interface*.
Import	Input parameter to an RFC. Input parameters are simple values (such as string or number) or structures.
RFC Adapter Service	SAP Adapter terminology for an association between a function module on an SAP system and an adapter service based on the RFC Adapter Service template on the SAP Adapter.
Structure	SAP data structure containing one or more fields. This can be thought of as a single row of a table.
Table	SAP data structure that is both an import and export to a function. Tables can be created and passed to an RFC. The function module can modify these tables and return them. The tables themselves are no different from relational database tables; they consist of a series of fields and rows of data for these fields.

Creating an RFC Adapter Service

Use the following procedure to create an RFC adapter service that makes a remote function call to a function module on the SAP system. This example shows how to create an RFC adapter service for RFC_FUNCTION_SEARCH.

Note: An SAP Adapter outbound call is an inbound call from the SAP system's point of view.

To create an adapter service for an RFC

1 Open the Developer.

2 Select **New** ▸ **Adapter Service**. Click **Next**.

3 Select **SAP Adapter** from the list of available adapter types. Click **Next**.

4 Select a previously created RFC connection that points to the SAP system that hosts the function module for which you want to create the adapter service. Click **Next**.

5 From the list of available templates, select RFC Adapter Service (synchronous). Click Next.

6 Enter a name and select a folder where the RFC adapter service should be stored. Click Finish.

7 On the Function Search tab, in the Function Pattern field, type all or part of the name of the function module for which you want to create an adapter service. Use pattern-matching characters if you are unsure of the complete name and want the SAP Adapter to search for several RFCs with similar names. For this example, enter `RFC_FUNCTION_*` in the Function Pattern field.

   Under Function Name, a list of RFCs that match the criteria you specified in the previous step is displayed.

   Select the name of the RFC for which you want to create an adapter service. For this example, RFC_FUNCTION_SEARCH. Leave all other fields at their default values.

8 Click Save on the Developer to save your RFC adapter service.

Testing the RFC Adapter Service

Use the following procedure to test the RFC adapter service you just created.

To test an RFC adapter service for a function module

1 In the Developer, open the RFC adapter service you previously created.

   On the Function Search tab, you will see value RFC_FUNCTION_SEARCH in the Function Name field.

2 On the toolbar, click Run.

3 Enter `RFC_*` into field FUNCNAME to specify input. Click OK.

   The results of RFC_FUNCTION_SEARCH will be displayed in the FUNCTIONS table on the Results page of the Developer.

After you create an RFC adapter service that executes a function module, you can create other services and clients that invoke the service.

Note: If an optional table is not passed to an RFC adapter service, no values will be returned for that table.

To ensure that the optional table gets returned from the SAP system, in your calling service you need to ensure that a value is set at the input signature of the RFC adapter service you want to call. In Developer, click Set Value to assign an (empty) value to the optional input field of the RFC adapter service.

For example, you call an adapter service named TCC_MSS_GET_ERRORLOG from a wrapper flow service. This adapter service contains an optional table called `LOGLIST`. To pass this table, click Set Value for this parameter.

If the empty table should only be the default value, deselect Overwrite Pipeline Value. With these setting, the optional table is always returned.

Posting an RFC-XML Document from an HTTP Client

There are several possible ways to invoke the RFC adapter service which will call a function module. It is important that you understand the namespace convention for services (see the *webMethods Integration Server Administrator's Guide*).

You can invoke the service illustrated previously by calling the following URL:

```
http://<Integration
Server>:<port>/invoke/<folder1>.<folder2>/<service>
```

To invoke the service

1 In an interactive scenario, you can simply call it from a browser the same way you call any other service on Integration Server. The input parameters must then be URL encoded, for example:
`http://localhost:5555/invoke/app/rfcGetStructureDefinition?TABNAME=USER01`

The response can then be rendered via an HTML template.

2 In a fully automated scenario your application would, however, send the request XML document to this service. A template for a valid request document for each function module can be generated from the Lookup menu by clicking the RFC-XML button. To invoke the RFC adapter service, you must post it in the HTTP body of request to the URL (as above). Your application must also set some keys in the HTTP Header:

```
POST /invoke/<your complete service name> HTTP/1.1
--> the service name to be called
Host: <xyz>
--> host of http client
Content-type: application/x-sap.rfc
--> you have to set the content type. When sending an RFC-XML
document, the content type should be application/x-sap.rfc

Cookie: ssnid=<4711>
--> This is optional. On the first connect Integration Server will
generate a session cookie. When you use this for the following
communication you will be assigned to the same Integration Server
session. If you do not use the session cookie a new Integration Server
session will be created with every HTTP request.

Content-Length: <the length of the HTTP Body>
```

The HTTP Body must consist of the XML document, encoded in RFC-XML. Depending on whether the call was processed successfully, you will get a response or an exception XML document. You can also force the SAP Adapter to invoke the function module with tRFC by setting a transaction ID in the Envelope of the XML document. However, you should bear in mind that only function modules without return parameters can be called with tRFC.

Tip: To simulate a raw post, you can use the service pub.client:http.

To use the pub.client.http service

1 Using Developer, navigate to the pub.client:http service.

2 Select Test4Run.

3 Specify the URL of the RFC-XML handler service. For example,

```
http://localhost:5555/invoke/app/rfcGetStructureDefinition
```

4 Specify Post as the method.

5 Paste the document into the String field of the Data section.

6 Add one entry in the field headers:

Name: Content-type

Value: application/x-sap.rfc

For more information about the pub.client:http service, see the *webMethods Built-In Services Reference Guide*.

webMethods

CHAPTER 6

Adapter Notifications

Overview

You can create RFCs on the SAP system that invoke services on the Integration Server. This allows SAP users to access the information that is available via the SAP Adapter. Before you can create an RFC that invokes a service, or send an IDoc to the SAP Adapter, you must configure the SAP system to have an RFC destination for an RFC listener running at the SAP Adapter. This enables the SAP system to send RFCs to the SAP Adapter. You must also configure the SAP Adapter to have an RFC listener that listens for RFCs from the SAP system.

After you have the SAP system and the SAP Adapter configured, you can create a function module on the SAP system that requests the execution of a service synchronously or you can send an IDoc to the SAP Adapter for further synchronous or asynchronous processing. To do so you also need to create a listener notification on the SAP Adapter. The listener notification indicates either what service the SAP Adapter is to execute when it receives a message from the SAP system or how to publish a corresponding document.

The SAP Adapter supports three types of notifications that can be assigned to an RFC listener.

- RFC listener notification (synchronous)
- ALE listener notification (synchronous)
- ALE listener notification (asynchronous)

If there is no listener notification assigned to the RFC listener, the received message will be forwarded to the routing listener. For more information, see Chapter 8, "Routing Messages Through the SAP Adapter".

Note: In this context, synchronous or asynchronous refers to the processing on the Integration Server. It does not indicate how the message was processed on the SAP system. However, in the case of an ALE listener notification, there is always a TID assigned to the message. In the case of an RFC listener notification, a TID will be present in the pipeline only for transactional RFC calls from the SAP system.

Components of a Listener Notification

In addition to the request (and reply) field selection, a listener notification contains the following fields:

- A flag that indicates if incoming IDocs should be tracked to later ALE monitoring from the calling system.
- A flag that indicates whether the Confirm event should be forwarded to the receiver.

Monitor IDocs Flag

All ALE listener notifications and routing notifications that have an ALE message type assigned support the Monitor IDocs feature. For more information, see "Using the ALE Monitoring Features Via the SAP Adapter" on page 172.

Forward Confirm Event Flag

All synchronous adapter notifications (the RFC listener notification, ALE listener notification, and the routing notification) support the forward Confirm event feature. This is a useful feature in scenarios where a transactional request is received by the SAP Adapter and then forwarded to an SAP system via tRFC, and where the client wants to achieve transactional behavior from end to end.

Note: This feature is not supported for an asynchronous adapter notification. For asynchronous adapter notifications, the Broker application that receives the publishable document from the SAP Adapter should determine how to proceed after successfully executing and committing the transactional request.

If the `Forward Confirm event` flag is activated, the SAP Adapter tries to forward the Confirm TID event, which is triggered by the tRFC protocol, to the same destination where the document went. What happens then depends on the transport used.

Transports that have an SAP system or other Integration Server as the destination confirm the TID on the destination system. This may become important in those cases where the final receiver of the message is another SAP system. For each tRFC it receives, an SAP system keeps an entry in a *check table* (ARFCRSTATE) to protect against duplicate processing of the same document. Only after the sender has indicated via Confirm TID that this document will never again be posted, the receiving system can safely remove this entry from its check table.

Up to version 4.6, the SAP Adapter would not forward this Confirm TID event, so the check table in the receiving SAP system would keep growing. This feature enables a clean up this check table in scenarios like:

SAP system A->SAP Adapter A ->Internet->SAP Adapter B->SAP system B or

external client->SAP Adapter->SAP system

if the external client uses the service pub.sap.client:confirmTID.

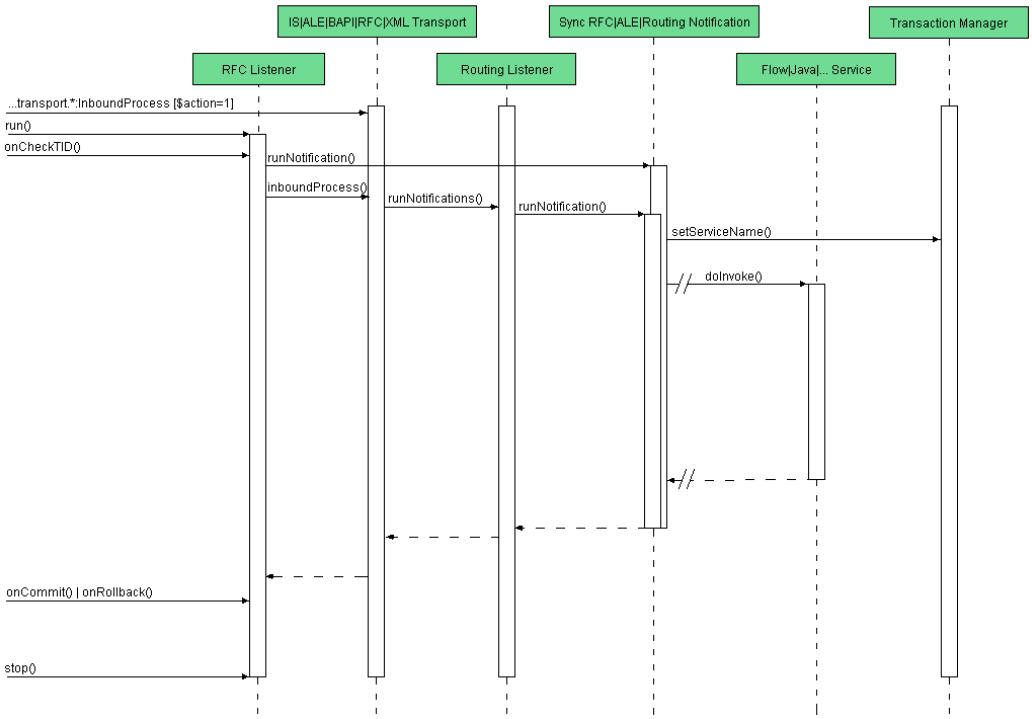
The following two simplified dynamic model diagrams show how the SAP Adapter handles requests differently depending on the setting of the *$action* parameter. They show a combined view of three possible scenarios:

- A listener notification is directly assigned to the RFC listener.
- No listener notification is assigned to the RFC listener, and the message is forwarded to the routing listener.
- The message is sent directly to the routing listener by calling one of the pub.sap.transport.*:InboundProcess services.

Action Equals 1

In the scenario where the value of *$action* equals `1` (Execute), the calling service name, or in case of an inbound transport, the matching confirm service, will be stored in the transaction store (setServiceName() call).

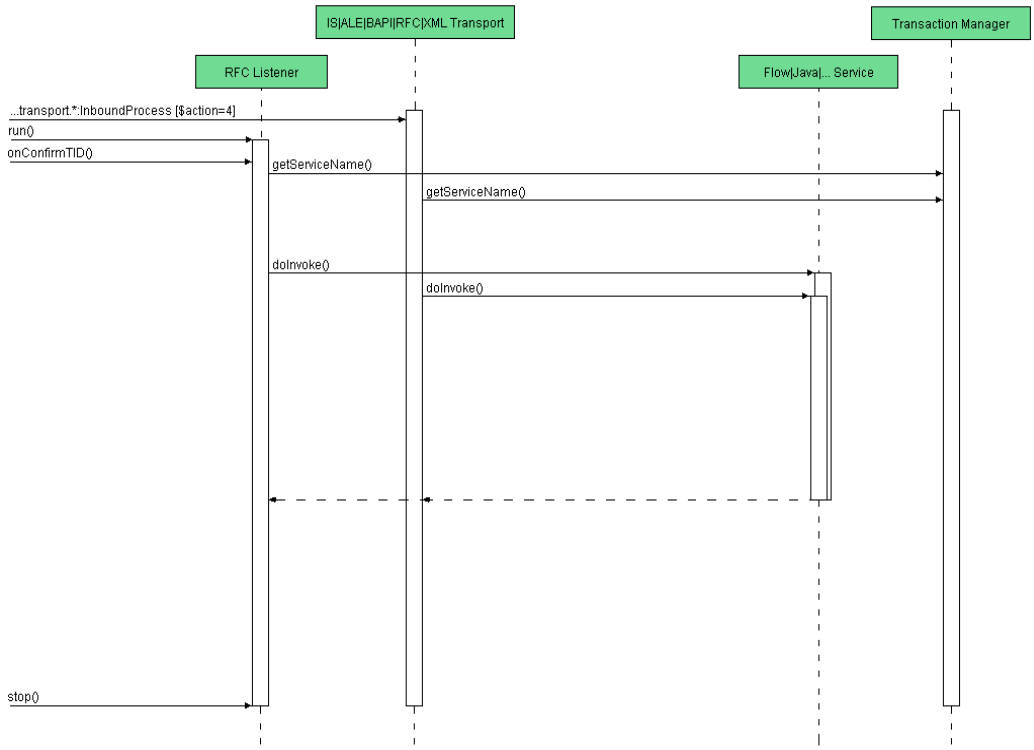
$action=1



Action Equals 4

In the scenario where the value of *$action* equals 4 (Confirm), the stored service name will be retrieved and the service will be called directly, bypassing the notification (getServiceName() call).

$action=4



Important! If you execute a routing notification that has a transport other than the Integration Server-assigned transport, the correct confirm service will be chosen and called without any additional coding needed. For all other scenarios, the assigned service will be called twice. You need to check the value of the *$action* field to correctly process the request.

Creating an RFC Destination on an SAP System

To enable your SAP system to issue remote function calls (RFCs) for services on an Integration Server, you must define an RFC destination on the SAP system. Each SAP system has a single RFC destination for an RFC listener defined at the SAP Adapter that identifies where the SAP system sends all RFCs that invoke services on the Integration Server.

Use the following procedure to configure the SAP Adapter listener as a registered RFC destination on the SAP system.

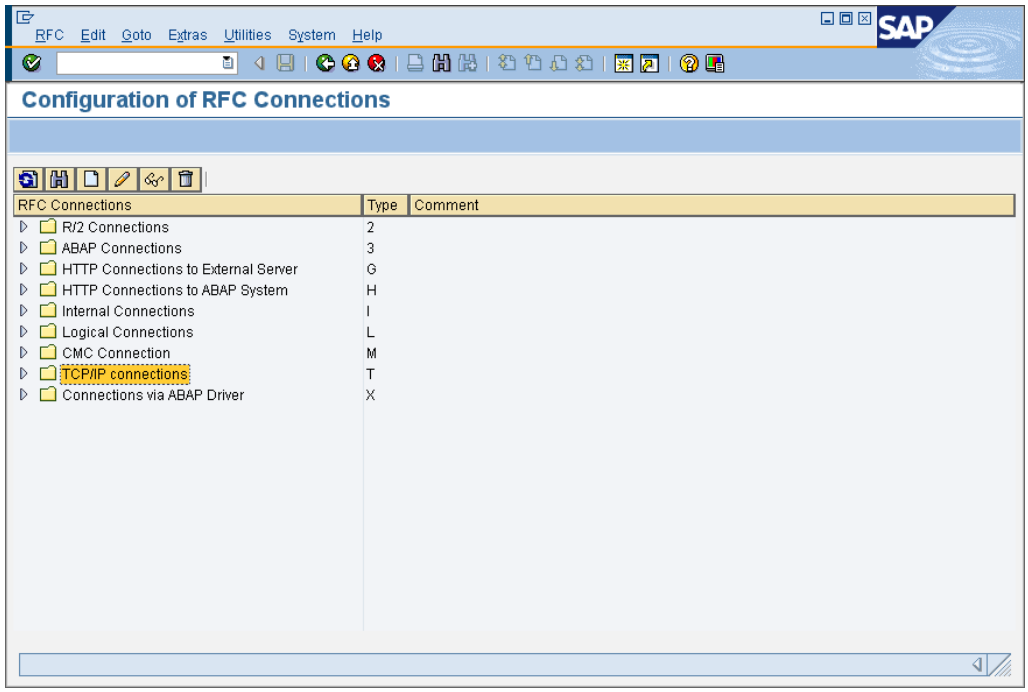
Note: You must have the proper authorizations on your SAP system to add an RFC Destination. If you do not have this authorization, have your SAP administrator perform the following steps.

To register SAP Adapter listener as an RFC destination

1 Use the SAPGui to login to the SAP system.

2 Select Administration ▸ System Administration ▸ Administration ▸ Network ▸ RFC Destinations (SM59).

3 Select TCP/IP connections.



4 Select **Create**.

5 In the **RFC Destination** field, type a name that will meaningfully identify both the SAP Adapter and the SAP system itself. For example, if the SAP system is named CER and the Integration Server is named IS, name your RFC destination ISCER. You will need to re-enter this name several times during the course of this section, so keep it simple and memorable.

Important! This field is case sensitive. It is recommend that you pick a name that is all UPPERCASE characters.

6 Enter `T` in the **Connection type** field (destination type TCP/IP).

7 Enter a description in the **Description** field that differentiates this SAP Adapter from any other.

8 Select **Save** from the toolbar or select **Save** from the Destination menu.

9 Select **Registration** as the **Activation Type**.

10 In field **Program ID** type the name of your RFC destination from step 5. Enter it *exactly* as you did in step 5. This is also a case sensitive field.

11 Select **Save** from the toolbar or select **Save** from the Connection menu.

12 Select **Gateway Options** from the Destinations menu.

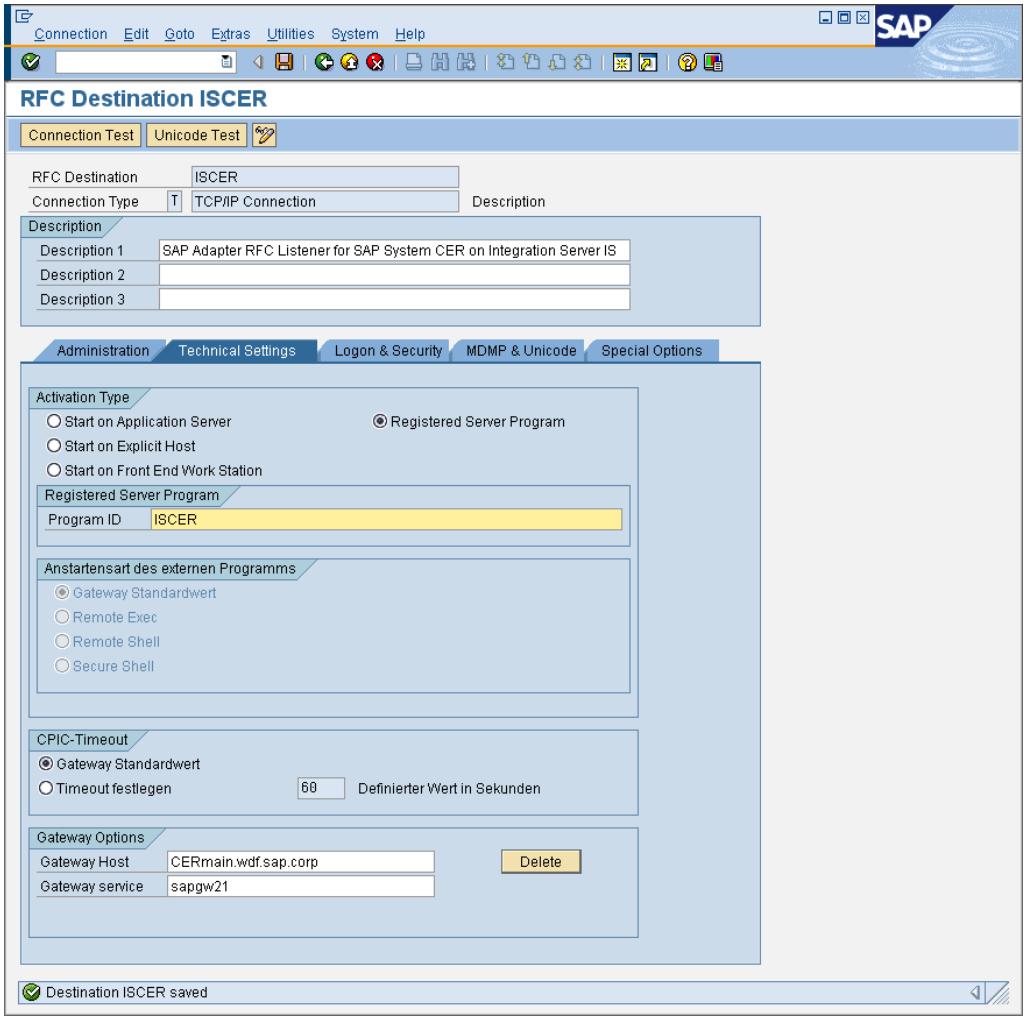
13 Enter *`SAP system application server`* in the **Gateway host** field.

14 Enter `sapgw`*`nn`* (where *nn* is the SAP system number) in the **Gateway service** field.

Note: This guarantees that you can access the RFC Server from all SAP application servers.

15 Select **OK**.

16 Select **Save**.



Remain on the current screen while you complete the steps for creating an SAP Adapter listener.

Listeners

This section describes how to create, modify, and delete listeners.

Before you Configure New Listeners

To prepare to configure a new listener

1. Make sure that you have webMethods administrator privileges so that you can access the SAP Adapter's administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide*.
2. Start the Integration Server and the Administrator, if they are not already running.
3. Using the Administrator, make sure that the WmSAP package is enabled. To verify the status of the WmSAP package, see "Enabling and Disabling Packages" on page 44.
4. Using the Developer, create a user-defined package to contain the listener, if you have not already done so. For more information about managing packages, see "Package Management" on page 41 for details.

Configuring an RFC Listener

The SAP Adapter requires an RFC listener to listen for inbound RFC requests from an SAP system. Use the following procedure to create a listener on the SAP Adapter to respond to RFCs issued by the SAP system.

To create an RFC listener on the SAP Adapter

1. In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.
2. In the SAP Adapter menu, click Listeners.
3. Select Configure new listener and then select RFC listener from the list of available listener types.

4 Complete the following fields on the Configure Listener Type screen (leave all other fields at their default values):

Field	Description/Action
Package	The package in which to create the listener. You must create the package using the Developer before you can specify it using this parameter. For general information about creating packages, see the *webMethods Developer User's Guide*. Note: Configure the listener in a user-defined package rather than in the adapter's package. See Chapter 3, "Package Management" for other important considerations when creating packages for the SAP Adapter.
Folder Name	The folder in which to create the listener.
Listener Name	The name of the new listener.
Program ID	The Program ID that you specified when creating the corresponding RFC destination on the SAP system. This field is case sensitive.
Gateway Host	Gateway Host for accessing your SAP system. This must be exactly the same parameter as you chose for the corresponding RFC destination in the SAP system.
Gateway Service	The Gateway Service. This corresponds to your SAP system number. If your SAP system number is "01" then your gateway service is "sapgw01." Again, this must be exactly the same parameter as you chose for the corresponding RFC destination in the SAP system.
Number of Threads	The number of simultaneous incoming RFCs that this listener can handle.
Repository Server	The outbound connection alias is used as a repository for function interfaces and structure definitions of inbound calls. This way it is possible to use RFCs even if they are not defined in the calling system.
SNC Enabled	Determines whether this server should use SNC. Default: No.

Field	Description/Action
SNC Quality of Service	SNC Quality of service, possible values: ■ Use global build-in default settings ■ Plain text, but authorization ■ Each data packet will be integrity protected ■ Each data packet will be privacy protected ■ Use maximum available security
SNC Name	Your own SNC name if you do not want to use the default SNC name. This is the name you chose when generating a PSE.
Authorization Service	Called by the RFC listener to check for authorization. To provide application specific authorization handling the user might provide a service here that implements specification pub.sap.listener:listenerAuthorizationCheck. This service has to return "Granted" in field "access" if the request should be accepted. If no service is specified access will be granted. If an exception happens during execution of this service or a different string than "Granted" will be returned the access will be denied. See "pub.sap.listener:listenerAuthorizationCheck." on page 269 for information about the service specification.
Unicode Listener	Enables or disables the listener to listen to a Unicode system. Click Yes, if your SAP system is a Unicode system.
RFC Trace	Whether you want RFC tracing enabled or disabled. For a production system, select Off. When you select On, SAP Adapter collects tracing in rfc*.trc files in directory *webMethods_directory*\Integration Server\. For detailed information about logging, see Chapter 10, "Logging and Monitoring". For information about changing the directory for the trace files, see Appendix B, "Server Configuration".

Field	Description/Action
Log transaction status	This switch can be set to `On` or `Off`. If set to `Off`, then the processing logs will not be saved, although a transaction will be created (or maintained), and the transaction can be monitored later on in the transaction list. **Tip!** Setting this switch to `Off` reduces the amount of disk space needed and the time it takes to log the transaction status. It still allows you to check the current transaction status on the SAP Adapter.
Store message body	This switch can be set to `On` or `Off`. If set to `Off`, then the message body of the incoming document will not be stored to disk, although a transaction will be created (or maintained), and the transaction can be monitored later on in the transaction list. **Tip!** Setting this switch to `Off` reduces the amount of disk space needed and of course the time it takes to persist the message body. It still allows you to track the message status on the SAP Adapter.

Important! An RFC listener must be able to login to the SAP system automatically when it starts up. Additionally, if there are no metadata yet for called function modules in the cache, there must be a log in option to the Repository System as well. Otherwise, it will fail to start up or return errors when receiving incoming RFCs.

5 Select **Save Listener** to commit these settings.

Enabling Listeners

After you have configured notifications, you must enable the listener so that the associated notifications will communicate appropriately with the listener at run time. You enable the listeners using the Administrator.

The **Status** column indicates the readiness of the listener. If the status is **Succeeded**, the listener is ready to be enabled. If the status is **Failed**, an error occurred during startup. If an error occurs during startup, the state will not change to "Enabled" when refreshing the page. Errors at this stage typically indicate a problem with either the listener configuration or the network. Review the listener settings and check the network.

For more information on configuring listeners and notifications, see the sections "Configuring an RFC Listener" on page 87 and "Configuring Listener Notifications" on page 98.

Note: When you reload a package that contains enabled listeners, the listeners will be enabled automatically when the package reloads. If the package contains disabled listeners, they will remain disabled when the package reloads.

To enable a listener

1 In the Adapters menu in the navigation area of the Administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners. The Listeners screen appears.

3 Select Enabled from the list in the State field. The Integration Server Administrator enables the listener.

The state changes to "Pending enabled". After refreshing the Listeners page, you should see the state changed to "Enabled".

After a listener is enabled, a connection exists between the SAP Adapter and the SAP system.

Tip! The Enable all suspended link helps you change the state quickly for multiple listeners.

Testing the RFC Listener

Use the following procedure to verify that the SAP system can successfully issue a remote function call (RFC) to the SAP Adapter.

To test the RFC listener

1 Toggle back to your SAPGui session. If your screen does not contain a Test Connection toolbar button, take the following steps.

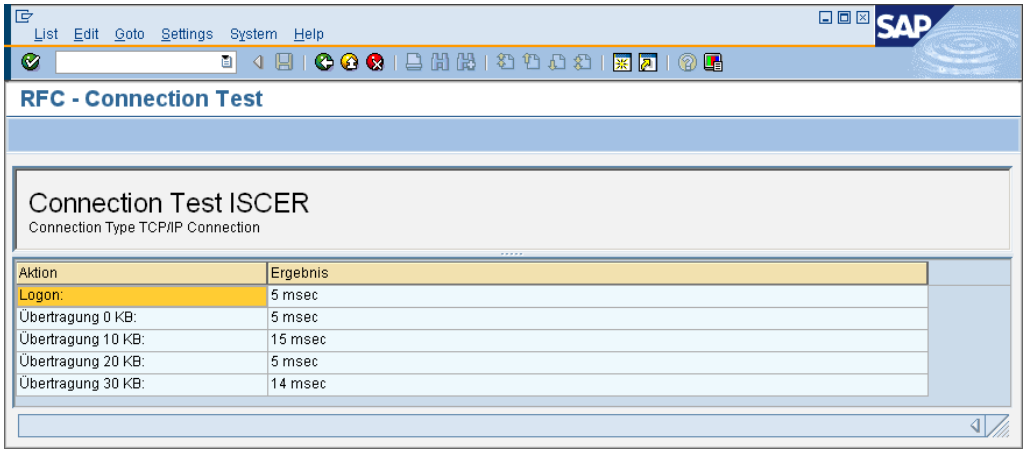
a Select Administration ▸ System Administration ▸ Administration ▸ Network ▸ RFC Destinations (SM59).

b Open the TCP/IP connections folder.

c Select the RFC destination you previously created.

2 Select the Test Connection toolbar button.

- If the SAP system can successfully connect to the SAP Adapter RCF listener, it will display connection information as shown below.
- If you receive an error message, review the steps for creating an RFC Destination and creating an RFC listener to verify your configuration settings.



Viewing Listener Parameters

You can view a listener's parameters from the Administrator or from the Developer. You also can view the notification order of a listener.

To view a listener's parameters using the Administrator

1 In the Adapters menu in the navigation area of the Administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners.

3 On the Listeners page, click the View icon for the listener that you want to see.

The View Listener screen displays the parameters for the listener. For descriptions of the listener parameters, see "Configuring an RFC Listener" on page 87.

4 Click Return to SAP Adapter Listeners to return to the main listeners screen.

To view a listener's parameters using the Developer

1 Start the Developer if it is not already running.

2 From the Developer navigation area, open the package and folder in which the listener is located.

3 Click the listener you want to view.

The parameters for the listener appear on the Listener Information tab. For descriptions of the listener properties, see "Configuring an RFC Listener" on page 87.

To view the notification order of a listener

1 In the Adapters menu in the navigation area of the administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners.

3 On the Listeners screen, click the View icon  for the listener that you want to view.

The View Notification Order screen displays the order of the notifications for the listener. To change the notification order for the listener, refer to the procedure "To edit the notification order of a listener" on page 94 for detailed instructions.

4 Click Return to SAP Adapter to return to the Edit Listener screen.

Editing Listeners

You use the Administrator to edit the listener in the following situations:

- If you need to select a newly configured connection, or if you need to change any listener properties you can update the listener parameters.
- If you need to change the order of the notifications that are associated with the listener, see the procedure on how "To edit the notification order of a listener" on page 94.

To edit a listener

1 In the Adapters menu in the navigation area of the Administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners.

3 On the Listeners screen, make sure that the listener is disabled before editing. To disable the listener, see "Disabling Listeners" on page 96.

4 On the Listeners screen, click the Edit icon  for the listener that you want to edit.

The Edit Listener screen displays the current parameters for the listener. Update the listener's parameters by typing or selecting the values you want to specify.

For descriptions of the listener parameters, see "Configuring an RFC Listener" on page 87.

5 Click Save Changes to save the listener and return to the Listeners screen.

To edit the notification order of a listener

1 In the Adapters menu in the navigation area of the administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners.

3 On the Listeners screen, make sure that the listener is disabled before editing. To disable the listener, see "Disabling Listeners" on page 96 for details.

4 On the Listeners screen, click the Edit icon  for the listener that you want to edit.

5 On the Edit Listener screen, click Edit Notification Order.

6 On the Edit Notification Order screen, use the Up and Down buttons to determine the processing order in which the SAP Adapter invokes the notifications.

Note: For better processing results, arrange your notifications from ascending to descending order starting with the most detailed notifications to the least detailed notifications. For more information on notifications and their filter criteria, see "Dependencies for Listener Notifications" on page 97.

7 Click Save Changes to save the notification order of the listener.

8 Click Return to Edit Listeners to return to the Edit Listener screen.

Copying Listeners

You can copy an existing listener to create a new listener with the same or similar properties without having to type or specify all properties for the listener. You copy adapter listeners using the Administrator.

To copy a listener

1 In the Adapters menu in the navigation area of the Administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners.

3 On the Listeners screen, click the Copy icon  for the listener that you want to copy.

The Copy Listener screen displays the current parameters for the listener that you want to copy. Name the new listener and edit any listener parameters as needed by typing or selecting the values you want to specify.

For descriptions of the listener parameters, see "Configuring an RFC Listener" on page 87.

4 Click Save Changes to save the listener and return to the Listeners screen.

Deleting Listeners

If you no longer want to use a listener, use the following instructions to delete the listener. You use the Administrator to delete listeners.

Important! If you delete an SAP Adapter listener, any notifications that are defined to use the listener will no longer work. You cannot change which listener a notification uses after the notification is configured. However, you can change the parameters for an existing listener. For instructions, see "Editing Listeners" on page 93.

To delete a listener

1 In the **Adapters** menu in the navigation area of the Administrator, click **SAP Adapter**.

2 In the **SAP Adapter** menu, click **Listeners**.

3 On the Listeners screen, make sure that the listener is disabled before deleting it. To disable the listener, see "Disabling Listeners" on page 96 for details.

4 On the Listeners screen, click  for the listener you want to delete.

The Integration Server deletes the listener.

Suspending Listeners

You can suspend listeners for an indefinite period of time. Suspended listeners cannot be edited or deleted.

Important! Suspending listeners for the SAP Adapter has the same effect as disabling them. For more information about disabling listeners, see "Disabling Listeners" on page 96.

To suspend a listener

1 In the **Adapters** menu in the navigation area of the Administrator, click **SAP Adapter**.

2 In the **SAP Adapter** menu, click **Listeners**.

3 On the Listeners screen, select **Suspended** from the list in the **State** field. The Integration Server Administrator suspends the listener.

The **Suspend all enabled** link helps you change the state quickly for multiple listeners.

When you suspend a listener, the action may not take effect right away. You may have to wait as long as the time specified in the Timeout parameter for the listener. If one or more messages appear on the queue within that time interval, the adapter may receive and process the first message.

Disabling Listeners

Listeners must be disabled before you can edit or delete them. You disable listeners using the Administrator.

To disable a listener

1 In the Adapters menu in the navigation area of the Administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners.

The Listeners screen appears.

3 Select Disabled from the list in the State field. The Integration Server Administrator disables the listener.

When you disable a listener, the action may not take effect right away. You may have to wait as long as the time specified in the Timeout parameter for the listener. If one or more messages appear on the queue within that time interval, the adapter may receive and process the first message.

Listener Notifications

The following sections provide instructions for configuring and managing SAP Adapter listener notifications. The SAP Adapter has three types of listener notifications that you can configure:

- RFC listener notifications (synchronous)
- ALE listener notifications (synchronous)
- ALE listener notifications (asynchronous)

For more information on how listener notifications work, see "Adapter Listeners and Listener Notifications" on page 31.

Before You Configure Listener Notifications

To prepare to configure a listener notification

1 Install the webMethods Integration Server and the SAP Adapter on the same machine. For details, see the *webMethods SAP Adapter Installation Guide*.

2 Make sure that you have webMethods administrator privileges so that you can access the SAP Adapter's administrative screens. See the *webMethods Integration Server Administrator's Guide* for information about setting user privileges.

3 Start the Integration Server and the Administrator, if they are not already running.

4 Using the Administrator, make sure that the WmSAP package is enabled. To verify the status of the WmSAP package, see "Enabling and Disabling Packages" on page 44.

5 Configure a listener using the Administrator. For more information on how to configure a new listener, see "Configuring an RFC Listener" on page 87.

6 Using the Developer, create a user-defined package to contain the listener, if you have not already done so. For more information about managing packages, see "Package Management" on page 41.

Dependencies for Listener Notifications

The following table lists other objects you must configure or tasks you must complete to use listener notifications:

Task		Use this tool...
1	Configure an adapter connection. For details, see "Configuring Adapter Connections" on page 56.	Integration Server Administrator
2	Configure an RFC destination on the SAP system. For details, see "Creating an RFC Destination on an SAP System" on page 84.	SAP system
3	Configure an RFC listener. For details, see "Configuring an RFC Listener" on page 87.	Integration Server Administrator
4	Select the appropriate notification template and configure the notification. For instructions to configure notifications, see "Configuring Listener Notifications" on page 98.	Developer
5	For asynchronous ALE messaging, you should create an Integration Server trigger that subscribes to the Integration Server document type that the SAP Adapter created with the asynchronous notification. Refer to the *webMethods Publish-Subscribe Developer's Guide* for more information about using triggers.	Developer
6	Enable the adapter notifications. For instructions to enable listener notifications, see "Enabling Listener Notifications" on page 104.	Integration Server Administrator

Configuring Listener Notifications

This section describes how to configure the three types of listener notifications.

Note: Routing notifications are used when no RFC listener notification has been defined. For more information about routing listeners, see Chapter 8, "Routing Messages Through the SAP Adapter".

RFC Listener Notification (Synchronous)

To configure an RFC listener notification

1 From the File menu, select **New**.

2 Select **Adapter Notification** from the list of elements and click **Next**.

3 Select **SAP Adapter** as the adapter type and click **Next**.

4 Select **RFC Listener Notification (synchronous)** from the template and click **Next**.

5 Select the appropriate **Notification Listener Name** and click **Next**.

6 Type a unique name for the synchronous listener notification and select the appropriate folder. Click **Next**.

7 Select a service and click **Next**.

8 Click **Finish**.

The Adapter Notification template creates the following items:

- An RFC synchronous listener notification
- Two Synchronous Document Types:
  - Synchronous Reply Document Type
  - Synchronous Request Document Type

Note: You cannot edit any fields or properties on the **Publications Properties** tab for the Synchronous Request and Reply Document Types. The Integration Server does not publish these document types.

9 In the adapter notification service editor, you can select the Adapter Settings tab at any time to confirm the following listener notification properties:

- Adapter name
- Adapter listener name
- Adapter notification template
- Adapter notification service—To edit this service, see "Editing Synchronous Listener Notification Services" on page 102.

10 Select the Function Search tab to verify or modify the following:

Property	Description
Function Pattern	All or part of the name of the function module on the SAP system. Enter a wildcard-like pattern for the function module for which you want create the notification. You can use exact patterns, or patterns with single or multiple wildcard characters.
Group Pattern	All or part of the name of the Function Group on the SAP system. Enter a wildcard-like pattern for the Function Group for which you want create the notification. You can use exact patterns, or patterns with single or multiple wildcard characters.
Function Name	The resulting function module name matching the provided pattern.
Function Description	The function module description, if it is available for that function module.
Group Name	The group name the function module belongs to.

11 Select the Request Field Selection tab to specify which fields should match the arriving message to run the notification.

Select the fields by selecting the appropriate boxes in the Use column. You can also select all fields by using the Check all rows icon  or de-select all fields by using the Uncheck all rows icon  .

12 Select the Reply Field Selection tab to specify which fields should match the returning message from the notification.

Select the fields by selecting the appropriate boxes in the Use column. You can also select all fields by using the Check all rows icon  or de-select all fields by using the Uncheck all rows icon  .

13 Select the Additional Settings tab to set the Forward confirm event flag. For more information about the Forward Confirm Event Flag, see "Forward Confirm Event Flag" on page 81.

14 Select the Permissions tab to manage the access control list (ACL) information. Use the drop-down menu to select each of the ACL types. For general information about assigning and managing ACLs, see the *webMethods Developer User's Guide*.

15 From the File menu, select Save (or Save All).

ALE Listener Notification (Synchronous)

To configure an SAP synchronous listener notification

1 From the File menu, select New.

2 Select Adapter Notification from the list of elements and click Next.

3 Select SAP Adapter as the adapter type and click Next.

4 Select ALE Listener Notification (synchronous) from the template and click Next.

5 Select the appropriate Notification Listener Name and click Next.

6 Type a unique name for the synchronous listener notification and select the appropriate folder. Click Next.

7 Select a service and click Next.

8 Click **Finish**.

The Adapter Notification template creates the following items:

- An ALE synchronous listener notification
- Two Synchronous Document Types:
  - Synchronous Reply Document Type
  - Synchronous Request Document Type

Note: You cannot edit any fields or properties on the **Publications Properties** tab for the Synchronous Request and Reply Document Types. The Integration Server does not publish these document types.

9 In the adapter notification service editor, you can select the **Adapter Settings** tab at any time to confirm the following listener notification properties:

- Adapter name
- Adapter listener name
- Adapter notification template
- Adapter notification service—To edit this service, see "Editing Synchronous Listener Notification Services" on page 102.

10 Select the **IDoc** tab to verify or modify the following:

Property	Description
IDoc type	Identifies the type of IDoc expected by the listener notification.
Cim type	The IDoc type extension (CIM type / customer extension type).
SAP system release	The IDoc release.
Old IDoc type 2	The IDoc version; unchecked for a new version 3 IDoc, checked for old version 2 IDocs (like in 3.1 SAP systems).
Monitor IDocs	Set to "On" to have the SAP Adapter link the IDoc packet's TID with the DOCNUMs of the IDocs in that packet so that later ALE IDoc Monitoring will be possible. Set to "Off" to prevent linking the TID with the DOCNUMS.

11 Select the Request Field Selection tab to specify which fields should match the arriving message to run the notification.

Select the fields by selecting the appropriate boxes in the Use column. You can also select all fields by using the Check all rows icon  or de-select all fields by using the Uncheck all rows icon  .

12 Select the Additional Settings tab to set the Forward confirm event flag. For more information about the Forward Confirm Event Flag, see "Forward Confirm Event Flag" on page 81.

13 Select the Permissions tab to manage the access control list (ACL) information. Use the drop-down menu to select each of the ACL types. For general information about assigning and managing ACLs, see the *webMethods Developer User's Guide*.

14 From the File menu, select Save (or Save All).

Editing Synchronous Listener Notification Services

The listener notification service is specified during the initial configuration of the synchronous listener notification. Afterwards you may need to change the service that is invoked. To change the notification service, complete the following steps.

Important! Before you select a different service, make sure that you have disabled the notification. When the notification is enabled, the new service is utilized. To disable the notification, see "Disabling Listener Notifications" on page 107.

To edit the notification service

1 From Developer, in the adapter notification service editor, select the Adapter Settings tab.

2 Open the Select Menu for that service.

3 Select a new service.

4 Click OK.

5 From the File menu, select Save (or Save All).

ALE Listener Notification (Asynchronous)

To configure an asynchronous ALE listener notification

1 From the File menu, select **New**.

2 Select **Adapter Notification** from the list of elements and click **Next**.

3 Select **SAP Adapter** as the adapter type and click **Next**.

4 Select **ALE Listener Notification (asynchronous)** from the template and click **Next**.

5 Select the appropriate **Notification Listener Name** and click **Next**.

6 Type a unique name for the ALE asynchronous listener notification and select the appropriate folder. Click **Next**.

7 Click **Finish**.

The Adapter Notification template creates the following items:

- An ALE asynchronous listener notification
- A Publish Document Type

In the adapter notification service editor, you can select the **Adapter Settings** tab at any time to confirm the following listener notification properties:

- Adapter name
- Adapter listener name
- Adapter notification template

8 Select the **IDoc** tab to verify or modify the following:

Property	Description
IDoc type	Identifies the type of IDoc expected by the listener notification.
Cim type	The IDoc type extension (CIM type / customer extension type).
SAP system release	The IDoc release.
Old IDoc type 2	The IDoc version; unchecked for a new version 3 IDoc, checked for old version 2 IDocs (like in 3.1 SAP systems).
Monitor IDocs	Set to “On” to have the SAP Adapter link the IDoc packet's TID with the DOCNUMs of the IDocs in that packet so that later ALE IDoc Monitoring will be possible. Set to "Off" to prevent linking the TID with the DOCNUMS.

9 Select the Request Field Selection tab to specify which fields should match the arriving message to run the notification.

Select the fields by selecting the appropriate boxes in the Use column. You can also select all fields by using the Check all rows icon  or de-select all fields by using the Uncheck all rows icon  .

10 Select the Permissions tab to manage the access control list (ACL) information. Use the drop-down menu to select each of the ACL types. For general information about assigning and managing ACLs, see the *webMethods Developer User's Guide*.

11 From the File menu, select Save (or Save All).

Enabling Listener Notifications

After you configure a listener notification, you need to enable it using the Administrator.

To enable a listener notification

1 In the Adapters menu in the Administrator navigation area, click SAP Adapter.

2 In the SAP Adapter menu, select Listener Notifications.

3 On the Listener Notifications screen, click No in the Enabled column for the listener notification you want to enable.

The Integration Server Administrator enables the listener notification and displays a  and Yes in the Enabled column.

Testing Listener Notifications

You can test listener notifications to ensure that you have configured them correctly.

To test listener notifications

1 Configure a listener using the Administrator. For instructions to configure a listener, see "Configuring an RFC Listener" on page 87.

2 Configure a listener notification using the Developer. For instructions to configure a notification, see "Configuring Listener Notifications" on page 98.

3 Enable the listener notification using the Administrator. For instructions to enable a listener notification, see "Enabling Listener Notifications" on page 104.

4 Enable the listener using the Administrator. For instructions to enable a listener, see "Enabling Listeners" on page 90.

5 On your SAP system, invoke a remote function call or send an IDoc to the RFC destination your RFC listener is listening to. The RFC listener will forward the received request to the matching notification.

Testing Publishable Document Types

You can test a publishable document type that is associated with an asynchronous notification using the run icon  in the Developer. When you test a publishable document type, you provide input values that the Developer uses to create an instance of the publishable document type. You also specify a publishing method (such as publish, publish and wait, deliver, or deliver and wait). Developer then publishes a document and displays the results of the publish in the Results dialog box. Testing a publishable document type provides a way for you to publish a document without building a service that does the actual publishing. If you select a publication action where you wait for a reply document, you can verify whether reply documents are received.

Note: Prior to running the *PublishDocument*, you need to make sure that you uncheck the Field must exist at run-time field on the *msgBody* property. To access this field, right-click *msgBody*, select Properties, and then select the Constraints tab.

When you test a publishable document type, the Integration Server actually publishes the document locally or to the Broker (whichever is specified).

For instructions to test a publishable document type, see the *webMethods Publish-Subscribe Developer's Guide*. Also, for a complete description of the envelope parameters located in the WmPublic folder, see the *webMethods Built-In Services Reference Guide*. The envelope parameters define the sender's address, the time the document was sent, password and certificate information, and other useful information for routing and control.

Viewing Listener Notifications

You use the Developer to view listener notifications.

To view a listener notification

1 In the Developer, expand the package and folder that contain the listener notification you want to view.

2 Select the listener notification that you want to view.

The Developer displays the configured listener notification in the adapter's Adapter Notification Editor.

Editing Listener Notifications

You use the Developer to edit listener notifications, both synchronous and asynchronous. When editing the listener notification, you can also edit the publishable document type associated with the asynchronous listener notifications or the request and reply document types that are associated with synchronous listener notifications.

To edit a listener notification

1 In the Developer, expand the package and folder that contain the listener notification you want to edit.

2 Select the listener notification you want to edit.

The Developer displays the configured listener notification in the adapter's Adapter Notification Editor.

3 Modify the values for the listener notification's parameters as needed. For detailed descriptions of the listener notification's parameters, see "Configuring Listener Notifications" on page 98 for the specific type of listener notification that you want to edit.

Note: Because listener notifications inherently depend on listeners, you cannot change a listener for a listener notification after you configure it.

To edit the document types

1 In the Developer, expand the package and folder that contain the document type that you want to edit.

2 Open the listener notification for the document that you want to edit.

3 Select the **Request Field Selection** or **Reply Field Selection** tab and modify the available values for the document type's parameters as needed. For detailed descriptions of the document type's parameters, see the appropriate procedure for that listener notification type.

Deleting Listener Notifications

If you no longer want to use a particular SAP Adapter listener notification, you can delete it by following the instructions in this section. You delete listener notifications, both synchronous and asynchronous, using the Developer.

Important! If you delete a synchronous listener notification, the associated request and reply document types are deleted automatically.

If you delete an asynchronous listener notification, the associated publishable document type is deleted automatically.

You cannot solely delete the document types associated with the synchronous and asynchronous listener notifications.

To delete a listener notification

1 In the Developer, expand the package and folder that contain the listener notification you want to delete.

2 Right-click the listener notification and click **Delete**.

Disabling Listener Notifications

You disable listener notifications using the Administrator.

To disable a listener notification

1 In the **Adapters** menu in the Administrator navigation area, click **SAP Adapter**.

2 In the **SAP Adapter** menu, select **Listener Notifications**.

3 On the Listener Notifications screen, click **Yes** in the **Enabled** column for the listener notification you want to disable.

The listener notification becomes disabled and **No** displays in the **Enabled** column.

Examples

Creating a Synchronous RFC Adapter Notification

The following tutorial explains how to assign inbound RFCs to services on the Integration Server. It describes an application in which the SAP system requests a service to retrieve information about a product.

Note: An SAP Adapter inbound call is an outbound call from the SAP system's point of view.

Creating a Function Module in an SAP System

Calling a service from an SAP system requires, at a minimum, a remotely callable function module with a defined signature (imports, exports, and tables). No ABAP coding, screen development, or other work is required.

Note: The following steps require that you have developer-level access on your SAP system and some basic knowledge of the ABAP Workbench and function modules.

To create a function module in an SAP system

1 Using the SAPGui, go to the ABAP Function Library. Select Tools ▸ ABAP Workbench (SE37).

2 Create a function group, for example: Z_FG01 (Menu Goto/Function groups/Create group).

3 Enter `Z_IS_PRODUCT` in field Function module. This is the name of your SAP product retrieval function.

4 Select Create.

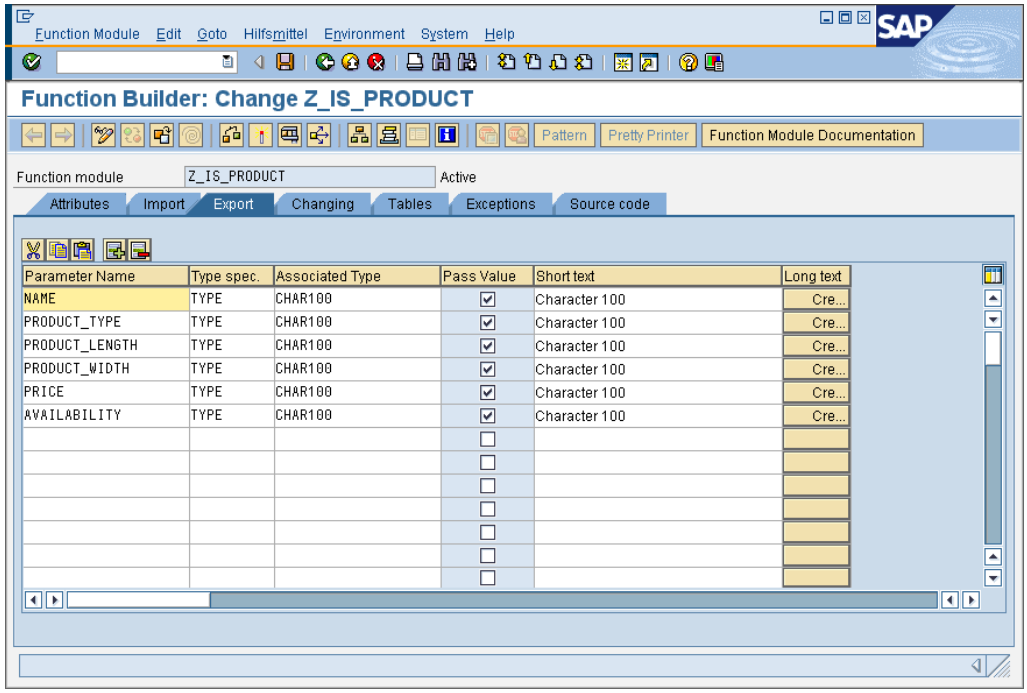
5 Complete the following dialogs in accordance with the policies governing your SAP development environment. The only aspect relevant to the SAP Adapter is the field Processing type. Select Remote-Enabled Module to allow this function to call externally to the SAP Adapter.

6 Define the import/export parameters of your function. Add an import named `sku`. You must provide an Reference Type field. Pick a character field with a length greater than 5. For this example, use `CHAR100`.

7 Add six exports: `name`, `product_type`, `product_length`, `product_width`, `price`, `availability`. Again, you must provide Reference Type fields for each of the exports. For this tutorial, use `CHAR100` for these parameters.

8 Mark Pass Value for all parameters.

9 Save your function module and activate it.



Creating an RFC Adapter Notification

You need to associate a service on the Integration Server with the inbound RFC. The following steps assign the app:getProductData service to inbound requests for Z_IS_PRODUCT.

To create an RFC adapter notification

1 Open the Developer and select New > Adapter Notification. Click Next.

2 Select SAP Adapter from the list of available adapter types. Click Next.

3 From the list of available templates, select RFC Listener Notification. Click Next.

4 Select a previously created RFC listener. The listener corresponds to the RFC destination you have created on the SAP system that hosts the function module for which you want to create the adapter notification. Click Next.

5 Enter a name and select a folder where the RFC adapter notification should be stored. Click Next.

6 Select the service that should be invoked by this adapter notification. (In this case, app:getProductData.) Click **Next** and then **Finish**.

7 On the **Function Search** tab, in the **Function Pattern** field, enter all or part of the name of the function module for which you want to create an RFC adapter notification. Use pattern-matching characters if you are unsure of the complete name and want the SAP Adapter to search for several RFCs with similar names.

For this example, enter `Z_IS_*` in the Function Pattern field.

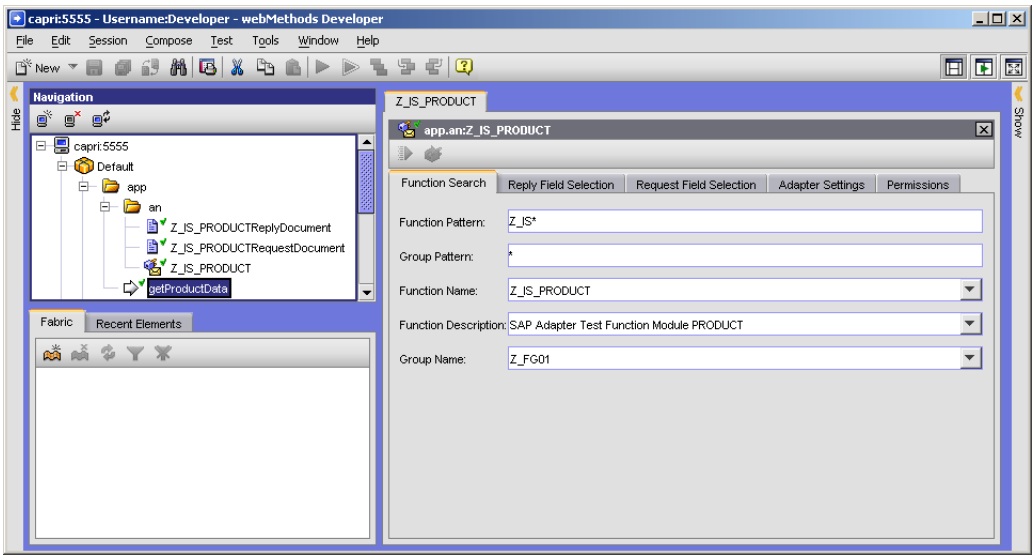
A list of RFC names that match the criteria is displayed.

8 Select the name of the RFC for which you want to create an adapter notification. For this example, use `Z_IS_PRODUCT`.

Note: If the RFC you expect to see is missing from the list, you might not have defined your RFC Connection correctly. Review the steps in "Configuring Adapter Connections" on page 56.

9 Click **Save** on the Developer to save your RFC adapter notification.

Leave all other fields at their default values.



Testing the Product Retrieval Function

Use the following procedure to test the RFC you just created.

To test an inbound RFC

Use the following procedure to test the product retrieval function:

1 Enable the RFC adapter notification.

2 Ensure that an RFC Listener is running by doing the following:

a In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

b In the SAP Adapter menu, click Listeners.

c Verify that the RFC listener you created for the RFC destination on the SAP system hosting `Z_IS_PRODUCT` shows status "Enabled", indicating that the listener is started and active.

If the status is "Pending enabled", refresh the listener screen until the listener has status "Enabled". If no listener appears in the list, refer to "Listeners" on page 87.

3 In an SAPGui session, go to the ABAP Workbench (SE37) to test your new function module.

4 Enter `Z_IS_PRODUCT` in field Function module and select Single test.

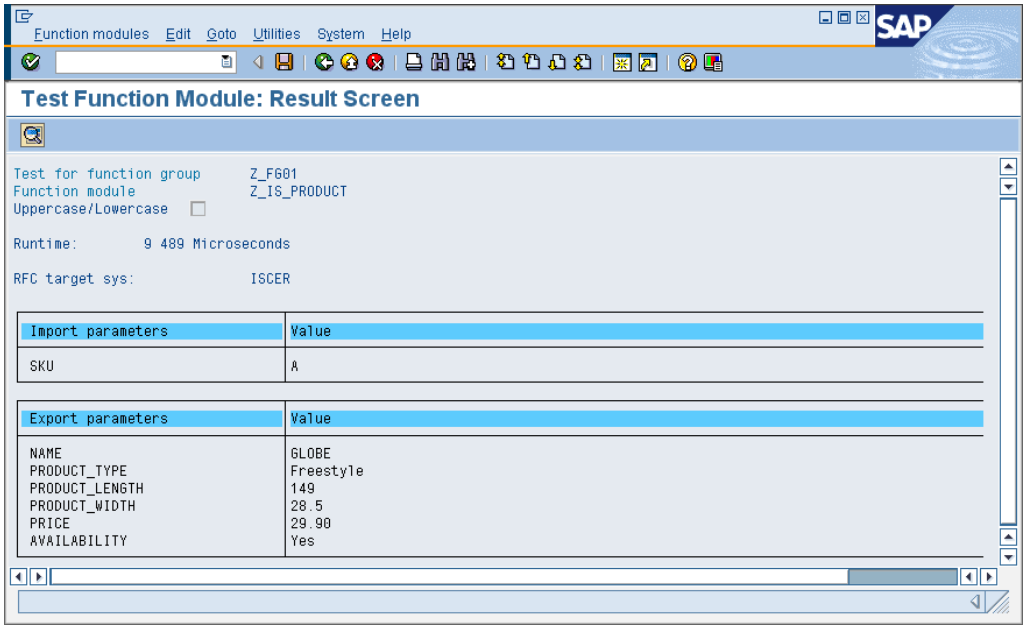
5 In the RFC target system field, type the name of your RFC Destination (in this example `ISCER`).

6 In the sku field, enter a SKU. For this example, the SKU's that are available are `A, B, C, D, or E`. Case is not important.

7 Execute (F8) the RFC by selecting the appropriate toolbar button or selecting Execute from the Function modules menu.

8 You will receive the product data in your exports list similar as shown in the following figure.

Note: If you receive an error from the SAP Adapter, make sure your RFC adapter notification is correct and that the app:getProductData flow service is available and functional.



Creating a Synchronous ALE Listener Notification

If you need to use the information contained in an IDoc in documents of another format, you build a service that "maps" the information from fields in the IDoc to the variables used by the other application. For example, to transfer a purchase order from an ORDERS02 IDoc to an EDI system, you create a flow service that maps information from the IDoc fields to fields within the EDI system's purchase-order document. Then, you define a routing notification that triggers this flow service.

To create a sample synchronous ALE listener notification, perform the following:

1 "Create an Empty Flow Service Called mapOrders02"

2 "Create an Asynchronous ALE Listener Notification for IDoc Type Orders02"

3 "Define the Input and Output Signatures for the mapOrders02 Service"

4 "Map Fields from Orders02RequestDocument to PurchaseOrder Document"

5 "Testing the Listener Notification from the SAP System"

Create an Empty Flow Service Called mapOrders02

Use the following procedure to create a flow service named app.idocs:mapOrders02 with Developer (this is the service that the synchronous ALE notification will invoke based on the routing rule created in the next stage).

To create an empty flow service

1 In the Developer, select the Default package and right-click to open the shortcut menu. Click **New**, then **Folder**, and name the new folder "apps".

2 Select the apps folder and create a subfolder called "idocs".

3 On the **File** menu, click **New** and create an empty flow service called mapOrders02. Save the service in the apps\idocs directory.

You will select this service when creating a listening notification.

Create an Asynchronous ALE Listener Notification for IDoc Type Orders02

This listener notification invokes the flow service that will perform the mapping. When you create the listener notification, select the service you have previously created for mapping the IDoc.

To create a synchronous ALE listener notification

1 From the File menu, select **New** > **Adapter Notification** and then click **Next**.

2 Select **SAP Adapter** as the adapter type and click **Next**.

3 Select **ALE Listener Notification (synchronous)** from the template and click **Next**.

4 Name the listener notification "Orders02", and select the app\idocs folder. Click **Next**.

5 Select the service you created in "Create an Empty Flow Service Called mapOrders02" on page 113, click **Next** and then click **Finish**.

6 Select the **IDoc** tab and change the **IDoc type** field to "ORDERS02".

7 Select the **Request Field Selection** tab and click the **Check all rows** icon  .

8 From the **File** menu, select **Save** (or **Save All**).

Define the Input and Output Signatures for the mapOrders02 Service

This service maps an ORDERS02 IDoc to a specific Purchase Order format. To do this, you need to define the input and output signatures for the service.

To define the input and output signatures

1 Open the mapOrders02 service and select the Input/Output tab.

2 For the Input field, open the Select dialog and select the apps\idocs\Orders02RequestDocument.

3 For the Output field, open the Select dialog and select the WmSAP\sample\sap\records\PurchaseOrder document.

Map Fields from Orders02RequestDocument to PurchaseOrder Document

After you define the input and output documents, you should map fields from the input to the output.

To map fields from ORDERS02 to PO

1 Add a map step.

2 Select the Pipeline tab and select one or more fields from ORDERS02 to map to fields in PO.

Testing the Listener Notification from the SAP System

Before you can test the listener notification, you must complete the procedures in "Setting Up the SAP System" on page 115.

To test the flow service you created, use the SAPGui to submit your sample IDoc to the SAP Adapter. Check the results of the service to ensure that the IDoc is being mapped correctly.

If you want to debug your service in the Developer, you can use the savePipelineToFile and restorePipelineFromFile services to capture a submitted IDoc

To test the listener notification

1 Use the SAPGui to submit an IDoc to your flow service. When the service executes, the savePipelineToFile operation will make a copy of the pipeline (which will include your IDoc) and save it to a file.

2 Delete the savePipelineToFile service and insert the restorePipelineFromFile service.

3 Select the Test ▸ Run command to execute the flow service. When it executes, the restorePipelineFromFile service will retrieve the copy of the pipeline containing your IDoc, which the remainder of the flow will operate on.

When you are finished testing, delete the restorePipelineFromFile service and save the finished flow.

Setting Up the SAP System

Note: You perform the following procedures on the SAP system. Due to differences among SAP systems versions and across platforms, these procedures can differ slightly from what you need to do. They should be used as a general guide to the steps you need to take.

To set up an SAP system to send IDocs to the SAP Adapter, use the SAPGui to perform the following steps:

1 "Create an RFC Destination"

2 "Define a Logical Port"

3 "Create a Partner"

4 "Create a Partner Profile"

5 "Create a Distribution Model for the Partner and Message Type"

Create an RFC Destination

You must create an RFC destination on the SAP system. For instructions on how to create an RFC Destination, refer to "Creating an RFC Destination on an SAP System" on page 84.

Define a Logical Port

The lower level networking requires that a system port number be associated with the RFC destination. The logical port identifies the port to which messages are sent. The logical port can only be used if an RFC Destination was previously created.

You can define a unique logical port using WE21 (alternatively, use the following menu path to do this: Main screen ▸ Tools ▸ Business Communication ▸ IDoc-Basis ▸ IDoc ▸ Port Definition).

1 Select the Transactional RFC tree item and click Create.

2 On the toolbar, click New Entries.

3 Either select your own descriptive port name or let the system generate one.

4 Enter the IDoc version you want to send via this port, the RFC destination you just created, and a short description of your logical port, and then save the information.

Create a Partner

A logical subsystem manages one or more RFC Destinations. You can create a partner (logical system) using SPRO_ADMIN (alternatively, use the following menu path to do this: Main screen ▸ Tools ▸ AcceleratedSAP ▸ Customizing ▸ Project Management).

1 Select SAP Reference IMG.

2 Expand the following nodes: Basis Components ▸ Application Link Enabling (ALE) ▸ Sending and Receiving Systems ▸ Logical Systems ▸ Define Logical System. (You can also use SALE and select the path described above, starting with Application Link Enabling (ALE).

3 Select Define Logical System.

4 Click New Entries.

5 Enter an informative name for your partner and provide a short description. After saving the partner information, assign it to a workbench request.

Create a Partner Profile

Use WE20 to create a partner profile (alternatively, use the following sequence to do this: Main screen ▸ Tools ▸ Business Communication ▸ IDoc-Basis ▸ IDoc ▸ Partner profile).

1 Select the LS (logical system) partner type in the tree view and click Create.

2 Enter in the Partner field the partner you created in "Create a Partner" on page 116, and save the partner profile.

3 Below the outbound parameter table control, click Insert entry.

4 Enter the message type of the IDoc, (for example: `MATMAS`).

5 Enter the logical receiver port you created before and enter the basic type of the IDoc, (for example: `MATMAS03`).

6 Save the outbound parameter.

7 Below the inbound parameter table control, click Insert entry.

8 Enter the message type of the IDoc, (for example: `MATMAS`) and the process code, (for example: `MATM`).

9 Save the inbound parameter.

Create a Distribution Model for the Partner and Message Type

After you define a partner and partner profile, you can create a distribution model that triggers the creation of a communication IDoc.

If you are using SAP system 4.5 or earlier, you can use BD64 to create the distribution model (alternatively, use the following sequence to do this: Main screen ▸ Tools ▸ Business Framework ▸ ALE ▸ Customizing).

1 Open the Cross-Application Components folder, then the Distribution (ALE) folder, then the Distribution Customer Model folder in the tree view.

2 Select Maintain customer distribution model.

3 Create a new model using Model ▸ Create.

4 Add a message type to your model, enter the sender in the dialog box (for example:, `CERCLNT800`), enter the receiver (your logical system), and the message type (`MATMAS`).

If you are using SAP system 4.6 or later, you can use BD64 or alternatively, the following procedure:

1 In the Main screen, select Tools ▸ AcceleratedSAP ▸ Customizing ▸ Project Management.

2 Select SAP Reference IMG.

3 Expand the following nodes: Basis Components ▸ Distribution (ALE) ▸ Modelling and Implementing Business Processes ▸ Maintain Customer Distribution Model.

4 Select Maintain Customer Distribution Model (BD64).

5 Change into the edit mode.

6 Select Create model view.

7 Enter a short text string and a technical name for your new model view.

8 Select your new model view in the tree Distribution Model, and select Add message type.

9 In the dialog box, enter the sender (for example: `CERCLNT800`), the receiver (your logical system), and the message type (`MATMAS`).

webMethods

CHAPTER 7

Generating Document Types

Generating Document Types for RFC Structure

This feature allows you to generate an RFC document type for an RFC structure defined at an SAP system.

Note: To do this, you must have the SAP Adapter plug-in installed on the machine that hosts the Developer.

To create a document type using the DDIC

1 Open the Developer.

2 On the File menu, select New.

3 On the New panel, select SAP Document Type and then click Next.

4 On the New SAP Document Type panel, select RFC and click Next.

5 Select the System ID and click Next.

6 Select the SAP Structure. If no structures have been cached, you must type the structure name into the field. Click Next.

7 Name the document type and select the folder where it will be placed. Click Finish.

Generating Document Types for IDocs

Create an IDoc Document Type Using the DDIC

You can create an IDoc document type using the metadata describing the IDoc structure that is available in the DDIC. This is the best approach for creating an IDoc document type.

Note: To do this, you must have the SAP Adapter plug-in installed on the machine that hosts the Developer.

To create a document type using the DDIC

1 Open the Developer.

2 On the File menu, select New.

3 On the New panel, select SAP Document Type and then click Next.

4 On the New SAP Document Type panel, select IDoc and click Next.

5 Select the System ID and click Next.

6 Select the IDoc type. If no IDocs have been cached, you must type the IDoc type into the field.

7 Complete the remaining fields as follows:

Field	Description/Action
Cim type	The IDoc type extension (CIM type / customer extension type).
SAP system release	The IDoc release.
Old IDoc type 2	The IDoc version; unchecked for a new version 3 IDoc, checked for old version 2 IDocs (like in 3.1 SAP systems).

Or optionally, leave them blank. Click Next.

8 Name the document type and select the folder where it will be placed. Click Finish.

Generating an IDoc Document Type from a DTD

Note: If you are using SAP system version 4.6A or higher, you can create a DTD for an IDoc from transaction WE60. (See your SAP documentation for procedures). If you want to generate a document type from a DTD that you have created, you must first create an XML file that defines a root element and points to this DTD. (Use the XML files that SAP provides as guides.) Use this XML file to build your document type.

To create a document type from an IDoc DTD

1 Open the Developer.

2 On the File menu, select New.

3 On the New panel, select Document Type and then click Next.

4 On the New Document Type panel, do the following:

a In the Folder tree, select the Folder into which you want to save the document type definition.

b In the Name field, type a name for the document type using any combination of letters, numbers, and/or the underscore character. (You might want to include the name of the IDoc in the name.)

5 Select Next

6 On the XML Format panel, select Local DTD or XML Document, and press Next.

7 From http://ifr.sap.com you can download the corresponding XML file or DTD file (or schema) to any directory your developer has access to.

8 Select Finish.

9 If your IDoc is earlier than Version 3, edit the document type to remove the "40" designation from the control header element. For example, change `EDI_DC40` to `EDI_DC`.

10 Click the Save button.

Generating a Document Type from a Sample IDoc

If you do not have access to an SAP system or if you do not have (or cannot create) a DTD for your IDoc, you will have to generate your document type from a sample document that you submit to the Developer at design time. If you have to use this option, you must obtain (or create) a comprehensive sample document before you begin building the service. The sample document should contain examples of all possible fields in the IDoc. (Fields that are not represented in the sample document, will not appear in the document type.)

To save a pipeline image capturing the IDoc

1 Open the Developer.

2 On the File menu, select New.

3 On the New panel, select Flow Service and then click Next.

4 On the New Flow Service panel, do the following:

a In the Folder tree, select the Folder into which you want to save the document type.

b In the Name field, type a name for the document type using any combination of letters, numbers, and/or the underscore character. (You might want to include the name of the IDoc in the name.)

5 Press Finish.

6 Now switch to the empty Flow Service you just created.

7 Click  on the Flow Pane toolbar, and select the pub.flow:savePipeline service. (If this service does not appear in the list, select Browse... to find it.) This service will copy the contents of the pipeline so that you can retrieve it in a later stage. (Later on the savePipeline step will be deleted from your flow again. Its purpose is simply to capture a copy of the IDoc— it is not a permanent part of your flow.)

8 Select the Pipeline tab.

9 Select the *$name* variable under Service In, and click  on the toolbar.

10 Type a name for the saved pipeline and select OK.

11 Click  to save the flow service.

12 Send your sample IDoc to this service as described below:

Use the SAPGui or the utility at /WmSAP/submitIDocXML.html to submit your sample IDoc or simply send an IDoc over HTTP to the routing listener. Specify sender, receiver, and msgType as specified for the routing notification that should invoke the app:mapOrders02 service. For more information on how to create a routing notification see "Configuring a Routing Notification" on page 132.

Make sure that the document you submit is a comprehensive example that contains all possible fields for that IDoc. When the routing listener receives this document, it invokes the flow service you created above, which captures the IDoc by making a copy of the pipeline. In the next step, you will retrieve the saved image of the pipeline.

Note: The pipeline image created by the savePipeline operation is stored in memory and can be recalled by any subsequent service. However, the image is not stored on disk. If the server is restarted, it will no longer be available. You can create a permanent copy by using savePipelineToFile instead.

Use the following procedure to retrieve the pipeline image you created:

1 In the Developer, create a new Flow the same way, you created app.idocs:mapOrders02

2 Select the Flow tab.

3 Select  on the Flow Pane toolbar and select the pub.flow:restorePipeline service. (If this service does not appear in the list, select Browse... to locate it.) This service will retrieve the contents of the pipeline you saved previously.

4 Set input field $name of this service to the name of the saved pipeline and select OK.

5 As the next step of this service insert pub.xml:xmlNodeToDocument (to be found also in the WmPublic Package), if the IDoc was sent to the SAP Adapter via http, or pub.sap.idoc:iDocToDocument (to be found in the WmSAP Package), if the IDoc was sent to the SAP Adapter from an SAP system via tRFC or from the sample page `/WmSAP/submitIDocXML.html`.

6 Click  to save this flow service.

7 Execute this service and then select the **Results** tab and locate the *document* variable. It should contain the data of your IDoc.

Now create an empty "Document Type" and copy the IDoc structure just captured:

a Select the root document defined within *document*. In this example, the root document is `ORDERS02`.

b Select **Edit ▸ Copy** to make a copy of the root document.

c Now switch back to the empty document type and mark the empty pane on the right hand side. Select **Edit ▸ Paste** to paste the structure of the captured IDoc into your document type.

d Click  to save this document type.

webMethods

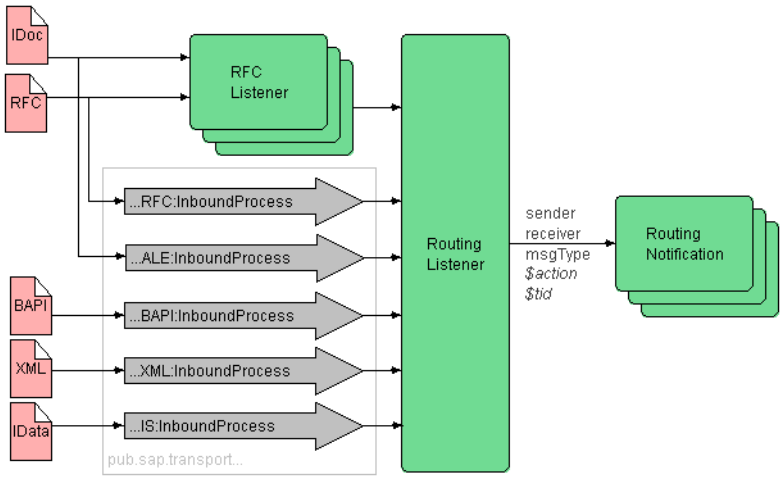
CHAPTER 8

Routing Messages Through the SAP Adapter

Introduction

This chapter describes how to route messages through the SAP Adapter.

When the SAP Adapter receives the IDoc or RFC, it matches the sender, receiver, and message type associated with the IDoc or RFC to its routing notifications. When it locates a matching routing notification, this is invoked to route the IDoc or RFC.



This chapter includes information about how to:

- Create and maintain routing notifications
- Update services that process routing notifications
- Map IDocs to other formats
- Use content based routing for IDocs and arbitrary XML documents as well as outbound mapping for IDocs

For specific information about how to:

Route...	Through the SAP Adapter, see...
RFCs	"Sending an RFC from an SAP System to the SAP Adapter" on page 138 and "Routing RFCs Through the SAP Adapter" on page 147
BAPIs	"Sending a BAPI from an SAP System to the SAP Adapter" on page 143 and "Routing BAPIs through the SAP Adapter" on page 103

Route...	Through the SAP Adapter, see...
IDOCs	"Sending IDocs with ALE from an SAP System to an SAP Adapter" on page 147 and "Routing IDocs through the SAP Adapter" on page 121
XML	"Routing Arbitrary XML Documents Through the SAP Adapter" on page 164

Overview

Normally, the RFCs and IDocs sent to the SAP Adapter from an SAP system are assigned to a specific service by an adapter notification. However, in some cases, you might want to route an RFC or IDoc through the routing listener.

The SAP Adapter includes a default routing listener that manages the routing of messages. It determines how and where to route a message based on routing notifications that you define. Each routing notification is associated with a transport that will be chosen and invoked when the routing notification is called.

When the routing listener receives a message, it performs a routing notification lookup. After locating the routing notification for the incoming message, the specified outbound transport gets invoked.

Routing notifications indicate how a message is to be processed. Each routing notification is uniquely identified by its sender, receiver, and message type. When the SAP Adapter receives a message, it performs a routing notification lookup to match the sender, receiver, and message type of the incoming message with the sender, receiver, and message type of the existing routing notifications.

If a matching routing notification is found, the SAP Adapter processes the message as the routing notification indicates. If a routing notification is not defined for the message, processing of the message will be aborted with an exception thrown.

The routing notification identifies a *transport*, which indicates how and where a message is to be routed. The transports you can specify allow you to route a message to a service, route an IDoc to an SAP system via tRFC, route an RFC/BAPI to an SAP system via RFC/tRFC, or post an IDoc-XML, RFC-XML, or bXML message to a URL.

Components of a Routing Notification

A routing notification contains:

1	Sender, receiver, and message type.
2	A flag that indicates whether the `Confirm` event should be forwarded to the receiver.

3	A flag that indicates if incoming IDocs should be tracked to later ALE monitoring from the calling system (Takes effect only if the request was received via a RFC listener).
4	How to route the message; that is, the transport.
5	Additional Parameters based on the selected transport.

Sender, Receiver, and Message Type

This information uniquely identifies a routing notification. The routing listener matches incoming messages to these fields to locate the routing notification to process the incoming message.

When a message is submitted, it must provide the routing listener with sender, receiver, and message type values. The sender, receiver, and message type values contain the following information:

Value	Description
sender	An arbitrary string that indicates who sent the message.
receiver	An arbitrary string that indicates the message's destination.
message type	Identifies the type of information the message contains (for example: a purchase order, a credit memo, an invoice, and so forth)

Note: For the Message types ORDERS and ORDRSP, a sample content based routing service is shipped with the SAP Adapter.

This replaces the sender/receiver information from the IDoc control record (SNDPRN and RCVPRN) by partner information from the E1EDKA1 segment. If this is not desired, disable this Content Based Routing from Adapters > SAP Adapter > Routing/Mapping.

You can write your own service for routing based on the fields which are used in your environment. You can use Content Based Routing for this purpose.

Refer to "Constructing an IDoc with the SAP Java IDoc Class Library" on page 186 and Appendix D, "Built-in Services" for the APIs to parse the IDoc.

Forward Confirm Event Flag

Routing notifications support the forward Confirm event feature as described at "Forward Confirm Event Flag" on page 81.

A call sequence in this case could be as follows:

1 The client creates a 24 alphanumeric GUID (or calls pub.sap.client:createTID to obtain one from the R/3 system).

2 The client sends the document to one of the InboundProcesses (for example an IDoc-XML to the pub.sap.transport.ALE:InboundProcess) and passes the TID along in the header field "`X-TID: <tid>`". The routing notification which processes this message should have the `Forward ConfirmTID Event` flag set to "true".

3 If and only if the client receives a return code 200 from the SAP Adapter in step 2, it calls the same InboundProcess again with `$action` set to 4 (`Confirm`) like this:

```
GET /invoke/pub.sap.transport.ALE/InboundProcess?$tid=<tid>&$action=4 HTTP/1.0
```

This achieves two things: The state in the SAP Adapter's transaction store is set to `Confirmed`, and the entry in the receiving SAP system's tRFC check table (ARFCRSTATE) is cleaned up. This may be important for tRFC performance, if the SAP system receives large numbers of documents.

4 If the client receives an error return code in step 2 or no response at all, it may later resubmit the same document (using the same TID) without needing to fear duplicate processing. Then, after one of the subsequent tries has finally succeeded, the TID can be `confirmed` as described in step 3.

Note: If the document is sent another time after the `Confirm` event has already been executed, this will lead to a duplicate document! The `Confirm` event should only be called if the client knows that the document has been processed without error and will therefore never resubmit this document again.

Routing Notification Order

A routing notification is executed when its Sender, Receiver, and Message Type values match an incoming document's values. When one or more of those values is a wildcard (*), then the routing notification is matched and executed based on its value precedence.

You can change this precedence order by modifying the notification order defined for the routing listener.

Considerations for Routing Notifications

For the routing listener, configure a default notification that processes all messages that are not associated with any other type of SAP Adapter notification. From the Administrator, edit the routing listener's notification order list and place the default notification last. For instructions on how to configure a notification, see "Routing Notifications" on page 132. For instructions on how to edit the notification order list, see "Routing Notification Order" on page 129.

Using Wildcards as Routing Criteria

You can use a single asterisk character (*), known as a *wildcard*, for the Sender, Receiver, and Message Type values in a routing notification. A wildcard matches any value that may be submitted with an incoming document. You can:

- Use a wildcard for the Sender parameter to serve as a "catch all" routing notification for a specific company. For example, suppose that you want to execute a particular routing notification for all documents from XYZ Company. You do not necessarily want to define routing rules for each message type that might be sent from XYZ Company, so you use the wildcard character (*) for the Message Type.
- Use a wildcard for all parameters (Sender, Receiver, and Message Type) to serve as a "last resort" routing rule. This routing rule will be executed when no other routing rules match the incoming document.

Example

Suppose that you have routing notifications set up with the following values and notification order:

	Sender	Receiver	msgType
1	CERCLNT800	partner1	ORDERS
2	*	partner1	ORDERS
3	CERCLNT750	*	ORDERS

If a message arrives with the following values:

Sender = CERCLNT800

Receiver = partner1

msgType = ORDERS

it will be routed according to the first routing notification as it has a higher ranking compared to the second routing notification.

Analogously, a message with the values:

Sender = CERCLNT750

Receiver = partner1

msgType = ORDERS

would be routed according to the second notification, not the third.

To edit the notification order of a listener

1 In the Adapters menu in the navigation area of the administrator, click SAP Adapter.

2 In the SAP Adapter menu, click Listeners.

3 Select the routing listener (wm.sap.internal.ls:routingListener) and change the State to Disabled.

4 Click the Edit icon  for the routing listener.

5 On the Edit Listener screen, click Edit Notification Order.

6 On the Edit Notification Order screen, use the Up and Down buttons to determine the processing order in which the SAP Adapter invokes the notifications.

7 Click Save Changes to save the notification order of the listener.

8 Click Return to Edit Listeners to return to the Edit Listeners screen.

9 Click Return to SAP Adapter Listeners to return to the Listeners screen.

10 Change the State of the routing listener to Enabled.

Routing the Message (Transport)

The transport indicates how the SAP Adapter will route the message (for example, route it to a service on the local or a remote host). The transport can be one of the following:

Transport	Description
IS	Routes the message to a service on the local Integration Server.
ALE	Routes an IDoc to an SAP system via tRFC.
RFC	Routes an RFC to an SAP system via RFC or tRFC.
XML	Posts a bXML, IDoc- or RFC-XML to a URL.
BAPI	Routes a BAPI to an SAP system via RFC or tRFC.

Based on the transport you select, the Notification Editor displays additional fields required by the selected transport.

Routing Notifications

Configuring a Routing Notification

Use the following procedure to create a routing notification.

To create a routing notification

1 Open the Developer and select **New** ▸ **Adapter Notification**. Click **Next**.

2 Select **SAP Adapter** from the list of available adapter types. Click **Next**.

3 From the list of available templates, select **Routing Notification**. Click **Next**.

4 Select the routing listener wm.sap.internal.ls:routingListener. Click **Next**.

5 Enter a name and select a folder where the routing notification should be stored. Click **Next**.

6 Select the service that should be invoked by this routing notification. Click **Next** and then click **Finish**.

7 Type in values for fields **Sender**, **Receiver**, and **Message Type** to assign the routing notification for corresponding inbound messages.

8 Select one of the following transports from the **Transport** field and specify the additional information that is specific to the type of transport you select. For information about the additional parameters that you need to supply for each transport, refer to the page indicated below.

Select this transport...	To...	Page
IS	Route the message to a service.	133
ALE	Route an IDoc to an SAP system	133
RFC	Route an RFC to an SAP system	134
BAPI	Route a BAPI to an SAP system	134
XML	Post a bXML, IDoc- or RFCXML or any arbitrary XML document to a URL.	135

Note: All but the IS transport will override the service you assigned to the routing notification in step 6.

9 Click **Save** on the Developer to save the routing notification.

10 Enable the routing notification through the Integration Server Administrator.

IS Transport

Use this transport to route messages to another service that executes on the local machine or on a remote Integration Server. The entire pipeline is submitted to the specified routing notification service. If you connect to a remote Integration Server, you should edit the assigned service of this routing notification and add a MAP step before the invoke step `pub.remote:invoke`, where you drop all unnecessary parameters from the pipeline so that the amount of data sent over the network is as small as possible. The IS Transport supports the Forward Confirm event feature. In case of *$action* set to `4` (Confirm event), the assigned service is called a second time. You can add pre- and post-processing steps to the pipeline if you wrap the service you want invoke and assign the wrapper service to the routing notification.

Tip! When sending an IDoc or an RFC between two SAP Adapters using the IS transport, if the receiving SAP Adapter is of a lower release than 6.5 and you are sending an IDoc, you need to add as an additional invoke step the service pub.sap.idoc:iDocToTables. This is because older SAP Adapter releases do not understand the new internal IDoc format, but instead expect the two tables `IDOC_CONTROL_REC_40` and `IDOC_DATA_REC_40`.

ALE Transport

Use this transport to route an IDoc to an SAP system via tRFC. Before you can route an IDoc to an SAP system, you must define the RFC connection to that SAP system. For instructions, refer to "Configuring Adapter Connections" on page 56.

Set the Configure ALE Transport parameters as follows:

Key	Value
serverName	Name of the SAP system to which you want to route the IDoc. Select an RFC connection alias from the drop down list. The drop down list contains the RFC connection aliases as defined on the SAP Adapter. For instructions on how to define an RFC connection to an SAP system, refer to "Configuring Adapter Connections" on page 56.

RFC Transport

Use this transport to route an RFC to an SAP system via RFC or tRFC.

Set the transport parameters as follows:

Key	Value
serverName	Name of the SAP system to which you want to route the RFC. Select an RFC connection alias from the drop down list. The drop down list contains the RFC connection aliases as defined on the SAP Adapter. For instructions on how to define an RFC connection to an SAP system, refer to "Configuring Adapter Connections" on page 56.

BAPI Transport

Use this transport to route an BAPI to an SAP system via RFC or tRFC.

Set the transport parameters as follows:

Key	Value
serverName	Name of the SAP system to which you want to route the BAPI. Select an RFC connection alias from the drop down list. The drop down list contains the RFC connection aliases as defined on the SAP Adapter. For instructions on how to define an RFC connection to an SAP system, refer to "Configuring Adapter Connections" on page 56.
Processing restrictions	

Processing restrictions: Some BAPIs can be called both synchronously and asynchronously. Callers can choose how they want to execute a call by specifying a transaction id in the XML header (see Appendix G, "Using BizTalk Envelopes with the SAP Adapter"). If you only want to allow one specific type of call (for example for performance reasons only asynchronous calls, or for administration reasons only synchronous calls). You can define restrictions using the drop down list:

no restrictions	Caller may decide to send both synchronous and asynchronous calls.

Key	Value	
	synchronous only	Caller may only send calls without a transaction ID. Messages with a transaction ID (asynchronous messages) are rejected and an XML error message is returned.
	asynchronous only	Caller may only send calls with a transaction ID. Messages without a transaction ID (synchronous messages) are rejected and an XML error message is returned.

XML Transport

Use this transport to post an SAP IDoc- or RFC-XML to a URL.

Set the transport parameters as follows:

Key	Value
url	URL to which you want to post the XML message.
xmlType	Select between the XML dialects SAP-XML, bXML, Values-XML, Arbitrary XML or SOAP XRFC (XRFC with SOAP envelope). If you select SAP-XML the content type is set to `application/x-sap.idoc` (respectively `.rfc`). Therefore the receiving server has to understand this content type. (This can be overridden using the following flag.) If Arbitrary XML is selected, the transport expects the XML document as string in the variable *xmlData*.
useTextXml	Flag that allows you to overwrite the content type of bXML, SAP-XML to text/xml, if set to true. The checkbox is disabled for other dialects.
useUTF8	Flag that allows you to force the renderers of bXML, SAP-XML to use the encoding utf-8, if set to true. The checkbox is disabled for other dialects.
useBAPI	Set this flag if you want to use the BAPI XML format. (This field is only active when using the bXML dialect.)
objectName	The name of the business object to which the call should be mapped. This value is case-sensitive. (Available only when using the bXML dialect and the BAPI format.)
bapiName	The name of the BAPI method, to which the call should be mapped. This value is case-sensitive. (Available only when using the bXML dialect and the BAPI format.)

Key	Value
httpUser	An optional parameter that allows you to specify a username for authentication on the remote Web system.
httoPassword	An optional parameter that allows you to specify a password for authentication on the remote Web system.

Tip! When sending an IDoc between two SAP Adapters via XML transport, use `http://<hostname>:<port>/invoke/pub.sap.transport.ALE/InboundProcess` as URL.

When sending an RFC to a second SAP Adapter over HTTP, use `http://<hostname>:<port>/invoke/pub.sap.transport.RFC/InboundProcess` as URL.

When the routing listener receives a message for which it cannot locate a routing notification, it logs the message in its transaction store and throws an Exception.

Disabling Routing Notifications

Use the following procedure to disable a routing notification. When you disable a routing notification, all routing service and rule information is preserved until you re-enable it. But incoming messages for this sender/receiver/msgType combination will not be routed by the SAP Adapter as long as the routing notification is disabled.

To disable a routing notification

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 In the SAP Adapter menu, click Listener Notifications.

3 Locate the routing notification that you want to disable. Under Enabled, click Yes until it turns to No.

Deleting Routing Notifications

When you no longer need a routing notification, you can delete it. Perform the following procedure to delete a routing notification.

Note: When you delete a routing notification, the SAP Adapter does not delete the service that is associated with the routing notification.

To delete a routing notification

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 In the SAP Adapter menu, click Listener Notifications.

3 Select the routing notification you want to delete and click the  icon in the Delete column.

Editing a Routing Service

When necessary, you can wrap the transport service into a customized service before assigning it to a routing notification. You can edit your wrapper service to incorporate additional operations before or after the transport delivers the message. For example, you can insert a post-processing or pre-processing service, or you can include some error handling operations after the transport service.

You can update the flow service to digitally sign a message before it is routed. Or, if you want to FTP an IDoc in EDI flat file format, you would insert the step pub.sap.idoc:encodeString. If the outbound data should be an XML document, you would use pub.sap.idoc:encode instead, or your own mapping service that creates the format of your choice.

Sending an RFC from an SAP System to the SAP Adapter

You can only route an RFC if you provide a valid SBCHEADER with your function call. To route RFCs through the routing listener, you must include a header table that contains information that the routing listener uses to route the RFC.

Note: Before the routing listener can receive an RFC, you must define an RFC listener for the RFC destination defined at the calling SAP system. For instructions, refer to "Listeners" on page 87.

To send an RFC to the routing listener, you must configure the SAP system to register the RFC listener as an RFC destination. For instructions on how to create an RFC Destination, refer to "Creating an RFC Destination on an SAP System" on page 84.

The SBCHEADER Table

You can write an ABAP wrapper that calls a function module with an RFC destination and add an additional table to the call statement.

Note: This table must not be defined in the function interface.

The name of the table must be SBCHEADER and it must have the following structure.

Component	Component type	DTyp	Length	Dec. places	Short text
NAME	SBCNAME	CHAR	32	0	SBC routing table, Keyfield
VALUE	SBCVALUE	CHAR	255	0	SBC routing table, value of Keyfield

This structure is defined in the SAP system starting with release 4.6A under the name SBCCALLENV. If you do not have this structure in your system, please define a similar one as above.

The header table can contain an arbitrary number of key/value pairs. If you want to pass additional keys to the SAP Adapter, you can define your own name/value pairs and insert them into the SBCHEADER table.

You would read out these fields inside a java module in the SAP Adapter with the following statements (case-sensitive):

```
IDataCursor idc = pipeline.getCursor();
IData sbcHeader = IDataUtil.getIData( idc, "sbcHeader");
if (sbcHeader != null)
{
    IDataCursor headerCursor = sbcHeader.getCursor();
    String property = IDataUtil.getString( headerCursor, "myProperty");
    headerCursor.destroy();
    System.out.println("Property was "+property);
}
idc.destroy();
```

If you want the SAP Adapter to invoke a routing notification to route the RFC, the header table must contain the sender and receiver for the RFC. When determining the routing notification to invoke, the routing listener uses the sender and receiver you specify in the header table and matches message types against the RFC name in place of a message type.

To have the SAP Adapter use a routing notification, include the following information in the header table:

Key	Value
`sender`	Name of the sender. This name should match the name of a sender in the routing notification you want the routing listener to use to route the RFC.
`receiver`	Name of the receiving partner. This name should match the name of the receiver in the routing notification you want the routing listener to use to route the RFC.

If you want to control the routing of an RFC from the SAP system directly (without requiring a routing notification on the SAP Adapter), you can include `transport` information in the header table. The transport indicates where the SAP Adapter is to route the incoming RFC. When the SAP Adapter receives an RFC that specifies the transport, it does not invoke a routing notification but directly passes the RFC to the specified transport. The transports that you can identify in a header table to dynamically route an RFC through the SAP Adapter are:

- Route the RFC to an Integration Server service
- Route the RFC to an SAP system
- Post the RFC-XML to a URL

The following describes the key/value pairs you must specify for each transport you can specify.

- To route the RFC to an Integration Server service, use the Integration Server service transport.

Key	Value
`transport`	`IS` This value identifies the transport. Specify the value exactly as specified above
`serverAlias`	Name of the Integration Server on which the service to invoke resides. If the service resides on the server that routes the message, specify (*local*). Otherwise, specify an alias for a remote server. For the routing to be successful, the server routing the RFC must have the defined alias for the remote server.
`folderName`	Name of the folder in which the service resides. The folder name is case sensitive; use the exact combination of upper and lower case letters.
`serviceName`	Name of the service to which to pass the RFC. The service name is case sensitive; use the exact combination of upper and lower case letters.
`valueScope`	Where you want the SAP Adapter to store the connection to the remote server. To save the connection in your own session, specify `SESSION`. Use `SESSION` when the work being performed requires state be maintained. To save the connection in a shared area, specify `GLOBAL`. Use `GLOBAL` when the work being performed is stateless.

- To route the RFC to an SAP system, use the RFC transport.

Key	Value
`transport`	`RFC` This value identifies the transport. Specify the value exactly as specified above
`serverName`	RFC Connection alias to which you want the RFC routed.

- To post the RFC-XML to a URL, use the XML transport.

Key	Value
`transport`	`XML` This value identifies the transport. Specify the value exactly as specified above
`url`	URL to which you want to post the RFC.
`xmlType`	The XML format you want the routing listener to use for the RFC. Specify SAP-XML if you want the RFC in an XML format that is compliant with the SAP XML specification. Specify `Values-XML` if you want the RFC in webMethods native XML format.
`httpUser`	User name to supply for a user name/password authentication challenge (optionally)
`httpPassword`	Password to supply for a user name/password authentication challenge (optionally)

For sending RFCs as bXML, please refer to "RFC Based XML Messages Using IFR Format" on page 148.

Example of Using an SBCHEADER Table

To test a function module with the function builder, write a wrapper module. The wrapper module calls your RFC function module.

Note: The SBCHEADER table cannot be added in the function builder directly and must not be part of the function interface of the module you want to call remotely.

For example, you want to invoke a function module called Z_DEMO_COPY. It echoes the input of type CHAR255 received in the INPUT parameter to the OUTPUT parameter. In the wrapper module Z_WRAPPER_DEMO_COPY, you would define an additional input parameter named DESTINATION and a table named SBCHEADER.

A wrapper for Z_DEMO_COPY could look like this:

```
FUNCTION Z_WRAPPER_DEMO_COPY.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"       IMPORTING
*"              VALUE(INPUT) TYPE CHAR255 OPTIONAL
*"              VALUE(DESTINATION) TYPE CHAR32
*"       EXPORTING
*"              VALUE(OUTPUT) TYPE CHAR255
*"       TABLES
```

```
*"              SBCHEADER STRUCTURE SBCCALLENV
*"----------------------------------------------------------------------
data:
    msg TYPE CHAR1024.

CALL FUNCTION 'Z_DEMO_COPY' DESTINATION destination
    EXPORTING
       input = input
    IMPORTING
       output = output
    TABLES
       sbcheader = sbcheader
    EXCEPTIONS
       no_input_given = 1
       communication_failure = 2 message msg
       system_failure = 3 message msg
       OTHERS = 4.

CASE sy-subrc.
WHEN 1.
  output = 'Exception received: NO_INPUT_GIVEN' .
WHEN 2.
  concatenate 'COMMUNICATION_FAILURE received:' msg into output separated
  by space.
WHEN 3.
  concatenate 'SYSTEM_FAILURE received:' msg into output separated by
  space.
WHEN 4.
  output = 'Exception received: OTHERS'.
ENDCASE.

IF sy-subrc <> 0.
  WRITE output.
ENDIF.

ENDFUNCTION.
```

The following example illustrates how to route the Z_DEMO_COPY function module to the SAP Adapter and invoke a routing notification to route the message. When testing the function module from the SAPGui, provide the following input values:

INPUT	Hello ...!
DESTINATION	ISCER
SBCHEADER	sender CERCLNT800 receiver DELL

The SAP system routes the RFC to the specified RFC-Destination ISCER (which corresponds to the name of the RFC listener). In the SAP Adapter, the routing listener invokes the routing notification that corresponds to the sender CERCLNT800, receiver DELL, and message type Z_DEMO_COPY. To determine how the SAP Adapter routes Z_DEMO_COPY function module, you would need to inspect the corresponding routing notification that the routing listener invokes.

The following example illustrates how to route the Z_DEMO_COPY function module to the SAP Adapter and directly invoking an outbound transport. When testing the function module from the SAP GUI, provide the following input values:

INPUT	Hello ...!
DESTINATION	ISCER
SBCHEADER	transport RFC serverName CERCLNT750

In this case, the SAP system routes the RFC to the specified RFC-Destination ISCER. In the SAP Adapter, the header table gets interpreted directly to determine how to route the RFC. The SAP Adapter routes the message to the SAP system known as CERCLNT750 on the SAP Adapter using transport RFC. Note that CERCLNT750 must correspond to the connection alias of an RFC connection that is configured in the SAP Adapter.

Sending a BAPI from an SAP System to the SAP Adapter

Overview

SAP systems communicate with the SAP Adapter on an implementation-level when calling BAPIs. This means, they try to call the implementing function module directly via RFC or, if asynchronous processing is used, they send an IDoc.

The SAP Adapter has a built-in converter to automatically rebuild the original object-based BAPI method call and represent it as an XML message. Therefore, the XML transport can handle these messages using the XML dialect *bXML*, to build a BAPI-style XML message.

To activate this function, you have to specify a routing notification that maps the BAPI call to the XML transport. The message types used in the routing notification for routing are based on the BAPI implementation. For messages received from an SAP system, this means:

- the RFC name of the BAPI implementation is used for synchronous calls
- the corresponding ALE message type is used for asynchronous calls

These message types can be found using the built-in BAPI browser. At the routing notification setup, you have to specify the name of the business object and BAPI to which the call should be mapped.

Setting Up a Routing Notification for the XML Transport

To enable this function, you will first have to setup an RFC Listener for the relevant SAP systems (see Chapter 6, "Adapter Notifications")

To create a routing notification with the following values set

1 Open the Developer and select New ▸ Adapter Notification. Click Next.

2 Select SAP Adapter from the list of available adapter types. Click Next.

3 From the list of available templates, select Routing Notification. Click Next.

4 Select the routing listener wm.sap.internal.ls:routingListener. Click Next.

5 Enter a name and select a folder where the routing notification should be stored.

6 Select any service that should be invoked by this notification. Click Next and then Finish.

Note: The service you selected in step 6 will only be used if the outbound transport that is chosen in your routing notification is "IS". For all other outbound transports, this selection will be overridden by a transport specific service.

7 Select XML in the Transport field to select the XML transport.

8 Select bXML Dialect for the xmlType field.

9 Enter a URL as destination of the XML call. To post the XML to another SAP Adapter, you can post the message to its BAPI inbound process. For this, you can post it to the URL.

```
http://<host>:<port>/invoke/pub.sap.transport.BAPI/InboundProcess.
```

10 Select Yes for the useBapi field.

11 Enter the object name in the objectName field. For example: "CompanyCode".

12 Enter the BAPI method that you want to use in the bapiName field. For example: “GetList”.

13 Optionally you can specify a username and password for user authentication on the remote host.

Note: You can select additional options for the XML transport configuration:

— Use text/xml as content type: Flag that allows to overwrite the content-type of bXML, SAP-XML to text/xml, if set to true.The checkbox is disabled for other dialects.

—Use utf-8 as encoding: Flag that allows to force the renderers of bXML, SAP-XML to use the encoding utf-8, if set to true.The checkbox is disabled for other dialects.

—SOAP XRFC can be selected as additional XML dialect. This is equivalent to XRFC (RFC-XML) with a SOAP envelope (higher than SOAP 1.1).

Dynamic Routing Using the XML Transport

It is not always necessary to specify routing notifications. If you want to call a BAPI from an ABAP report in your SAP system, you can also add the SBCHEADER parameter to your function module call. This parameter can be filled with key-value pairs that describe in detail how the message should be routed. At the moment, it is only possible to route BAPI-calls to the XML transport. To do this, use the following key pairs in the SBCHEADER table:

Name	Value
Transport	XML
url	The destination URL for the HTTP post operation
xmlType	bXML
httpUser (optional)	An optional parameter that allows you to specify a username for authentication on the remote web system
httpPassword (optional)	An optional parameter which allows you to specify a password for authentication on the remote web system
useBAPI	Set this value to YES if you want to use the BAPI XML format. If you omit this value or set it to something different than YES, the message will be sent as RFC based XML in a BizTalk XML envelope
objectName	The name of the business object, to which the call should be mapped. This value has to be case-sensitive.

Name	Value
bapiName	The name of the BAPI method, to which the call should be mapped. This value is case-sensitive
xsender	A value that should be put in the header `element <from> <address>`. See senderType for further details
senderType (optional)	An optional format descriptor, defining the sender address type. Default is `LogSys` for logical systems. Logical system names are automatically converted to an URN by an SAP defined schema. Alternatively, you can set senderType
xreceiver	A value that should be put in the header element `<to> <address>`. See receiverType for further details
receiverType (optional)	An optional format descriptor, defining the receiver address type. Default is `LogSys` for logical systems. Logical system names are automatically converted to an URN by an SAP defined schema. Alternatively, you can set receiverType
useTextXml	Flag that allows to overwrite the content-type to text/xml, if set to `Yes` (for SAPXML and bXML).
useUTF8	Flag that allows to force the renderers to use the encoding utf-8, if set to `Yes` (for SAP-XML and bXML).

For sending RFC based XML messages, only the first five parameters are supported. You can specify this parameter to your ABAP RFC as follows:

```
DATA header like SBCCALLENV occurs 1 with header line.
*... some code lines omitted
CALL FUNCTION 'BAPI_COMPANYCODE_GETLIST'
  DESTINATION 'ISCER'
IMPORTING
  RETURN = returnCode
TABLES
  COMPANYCODE_LIST = companyCodes
  SBCHEADER = header
        .
```

For a synchronous example, see "Calling a BAPI Synchronously from an SAP System" on page 189. For an asynchronous example, see "Calling a BAPI Asynchronously from an SAP System" on page 193.

Sending IDocs with ALE from an SAP System to an SAP Adapter

The routing listener receives IDocs from an SAP system if the tRFC (INBOUND_IDOC_PROCESS or IDOC_INBOUND_ASYNCHRONOUS) has no RFC adapter service associated with it. When the routing listener receives the IDoc, the sender, receiver and message type are extracted from the IDoc itself. The routing listener uses this information when determining where to route the IDoc.

Important! Before the SAP Adapter can receive the IDoc, you must define an RFC listener for the corresponding RFC destination defined at the calling SAP system. For instructions, refer to "Listeners" on page 87.

Note: For information about how to map the IDoc information into another format for use by another application, refer to "Mapping IDocs to Other Formats" on page 157.

Routing RFCs Through the SAP Adapter

Posting RFC Based IFR-compatible XML Messages

SAP Adapter supports the XML format for use with arbitrary RFC function modules. By using the new content-type `application/x-sap.busdoc`, which is also used for BAPIs, you can easily post RFC function calls to the SAP Adapter.

These XML messages must provide a BizTalk XML envelope (for further information, see Appendix G, "Using BizTalk Envelopes with the SAP Adapter"). The call of the corresponding business document for the RFC function can be put in the body of the BizTalk message.

By using the BizTalk XML envelope and this content-type, you also enable the support for the new application-specific XML error documents, which are defined in the SAP Interface Repository.

You can post to the generated proxy services URL for the RFC function modules, or to the RFC routing gateway

```
http://<server>:<port>/invoke/pub.sap.transport.RFC/InboundProcess
```

RFC Based XML Messages Using IFR Format

You can also generate XML calls for RFC function modules that are based on the IFR XML format. To do this, select bXML as the XMLType at the routing notification for RFC messages, and ensure that useBAPI is set to No.

Posting RFCs via FTP

Perform the following steps to FTP an RFC-XML document to the routing listener:

```
open <Integration Server host> <port of ftp listener>
username: Administrator
password: manage
cd ns/pub/sap/transport/RFC/InboundProcess
bin
put example.xrfc
```

The FTP client should check the return code to find out whether the document processed properly.

Note: If you are posting RFC-XML, the file name needs to end with `.xrfc`.

Posting RFCs in an E-mail Message

To send an RFC-XML document to the routing listener in an e-mail message, put the document into an attachment whose name ends with `.xrfc`. In the subject line of the e-mail specify pub.sap.transport.RFC:InboundProcess.

Routing BAPIs Through the SAP Adapter

Setting Up a Routing Notification and the BAPI Transport

Inbound messages in the BAPI-based XML format can be easily dispatched using routing notifications.

For each BAPI sent via XML to the SAP Adapter, a corresponding routing notification should exist. If no entry exists, the message will not be executed and the SAP Adapter will throw an exception.

The message type used in the routing notification for BAPIs is constructed from the concatenation of the business object name and the BAPI name, which are separated by a dot. (the syntax is <Business Object>.<BAPI method name>).

The only transport applicable to inbound BAPI calls is the BAPI transport.

Posting BAPI-based XML IFR-Compatible XML Messages

To apply the routing notification, the XML message should be posted to the SAP Adapter. The receiving service should be pub.sap.transport.BAPI:InboundProcess.

Execute an HTTP post operation to:

`http://<host>:<port>/invoke/pub.sap.transport.BAPI/InboundProcess`

In the HTTP body, an XML message specifying the BAPI-call as described above must be sent. You have to use the HTTP content type `application/x-sap.busdoc` when posting to the SAP Adapter.

```
POST /invoke/pub.sap.transport.BAPI/InboundProcess HTTP/1.0
Content-Type: application/x-sap.busdoc
User-Agent: Java1.1.8
Host: localhost:5555
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-length: 817
<?xml version=“1.0” encoding=“iso-8859-1”?>
  <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
    <header>
      <delivery>
        <message>
          <messageID>0A125F1315B3D24B0000001E</messageID>
          <sent>2000-06-20T09:58:00</sent>
        </message>
        <to>
          <address>urn:sap-com:logical-system:SAPSYS0001</address>
        </to>
        <from>
          <address>urn:sap-com:logical-system:SAPADA0001</address>
        </from>
      </delivery>
    </header>
    <body>
      <doc:CompanyCode.GetList xmlns:doc="urn:sapcom:
        document:sap:business" xmlns="">
        <CompanyCodeList>
          <item>
            <COMP_CODE></COMP_CODE>
            <COMP_NAME></COMP_NAME>
          </item>
        </CompanyCodeList>
      </doc:CompanyCode.GetList>
    </body>
  </biztalk_1>
```

Transaction Control

You can control whether the BAPI should be called synchronously via RFC or asynchronously via ALE by the `<referenceID>` element in the BizTalk header.

If the `<referenceID>` element is specified and contains a valid SAP transaction ID (TID), the BAPI outbound process automatically chooses the ALE format, otherwise, if the element is omitted, the message will be processed synchronously.

For both synchronous and asynchronous calls, technical processing errors are reported by an XML response document containing a fault descriptor.

BAPI XML Transaction Commit

When running a BAPI call with SAP Adapter in some cases the SAP system expects a *commit* command to execute the call. To commit the transaction directly over HTTP do the following:

1 Call lockSession with empty POST over HTTP

`http://<host>:<port>/invoke/pub.sap.client/lockSession`

2 Call the BAPI in the same session using HTTP

3 Commit the service with an empty POST using HTTP

`http://<host>:<port>/invoke/pub.sap.bapi/commit`

4 Release the session by sending an empty POST using HTTP

`http://<host>:<port>/invoke/pub.sap.client/releaseSession`

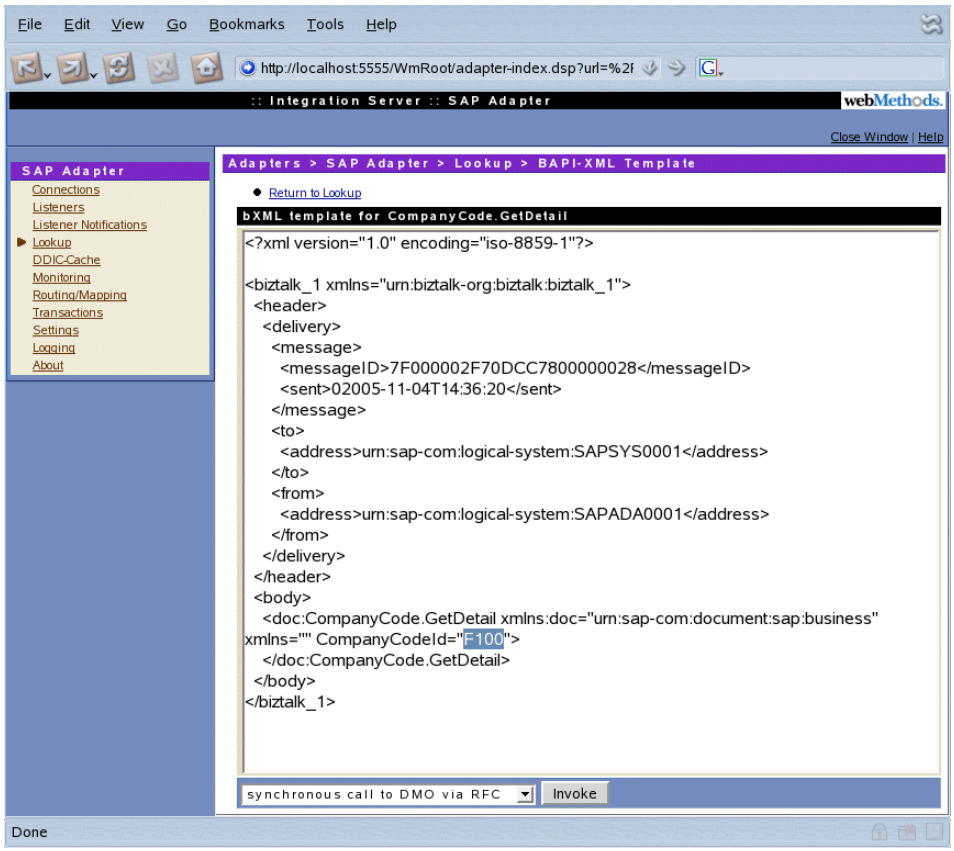
Note: With the HTTP requests in steps 2-4 you need to resubmit the cookie that you got with the response in step 1. Otherwise, the Integration Server cannot assign these requests to the same user session and the “commit” does not know which transaction to commit.

Synchronous Calls

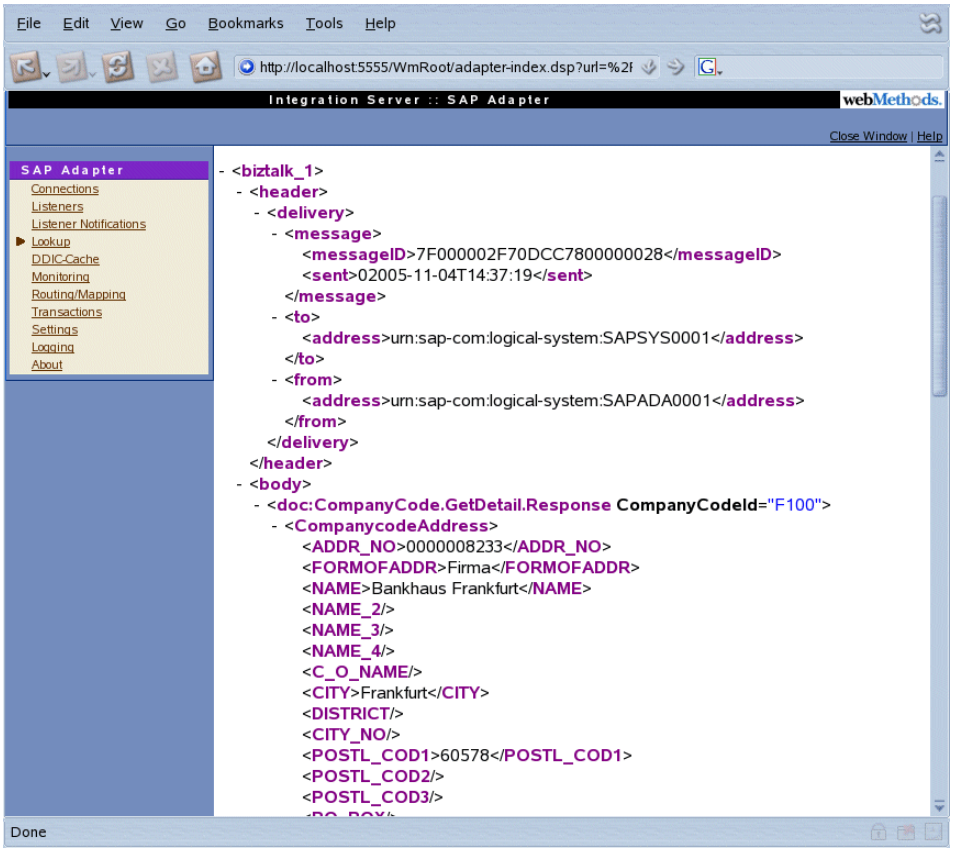
Application-level responses to synchronous XML requests are reported using a response business document in the HTTP response. The response business documents for synchronous calls are documented in the SAP Interface Repository and contain a serialization of all exporting and changing parameters of the BAPI or function module.

Example

- Sending a synchronous call to an SAP system



- It returns a response business document



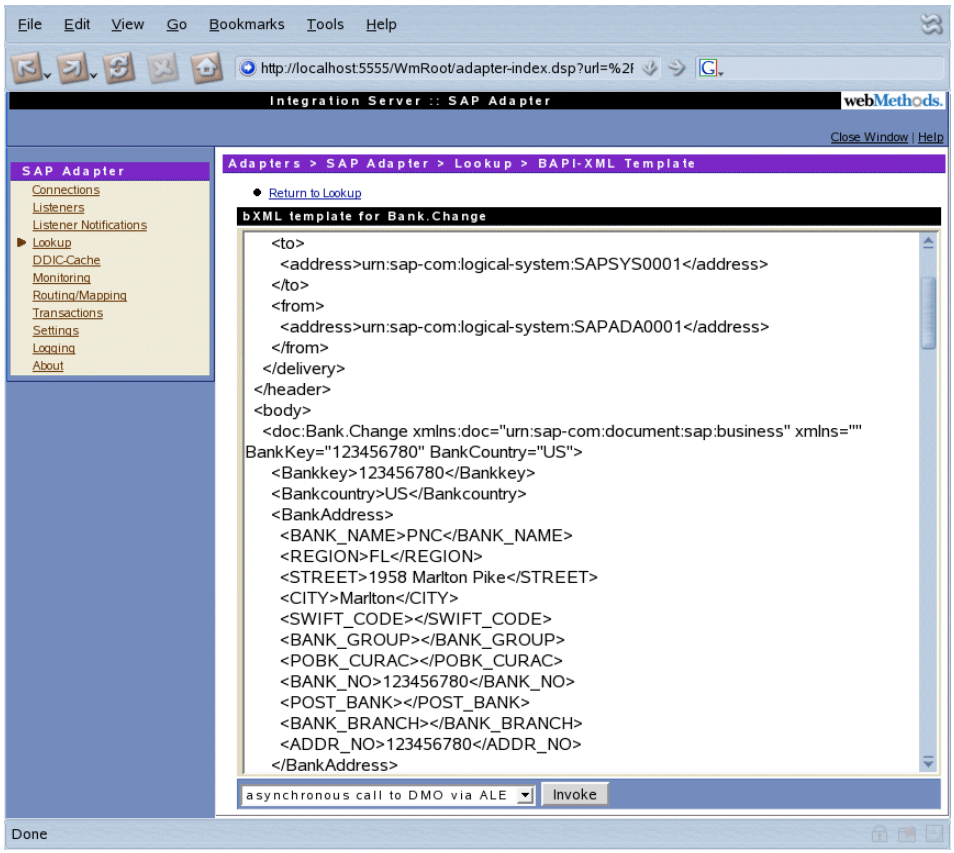
Asynchronous Calls

An asynchronous call can be executed by specifying a transaction ID in the BizTalk header of the XML document. The XML element used is the `<referenceID>`-element (see description of the BizTalk XML envelope) in the `<receiver>`-section of the BizTalk header.

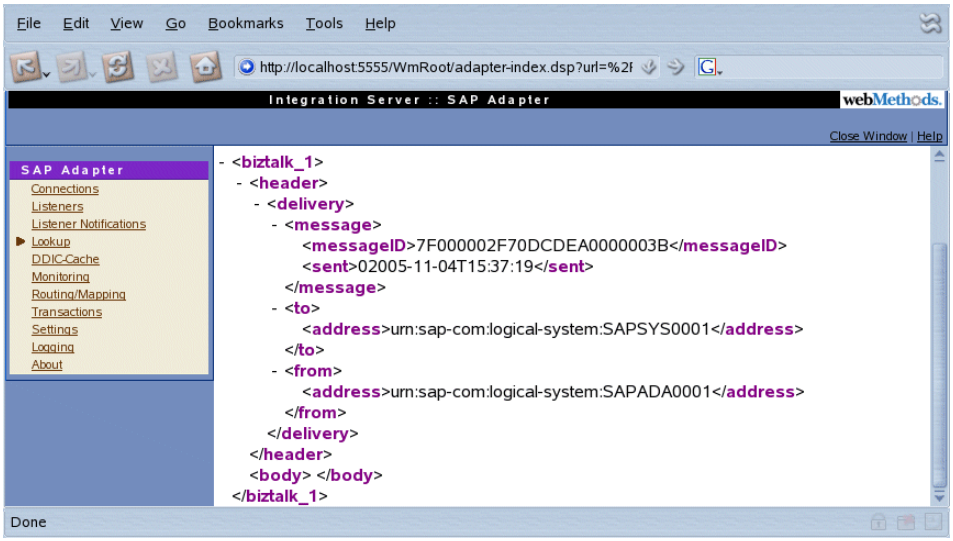
A transaction ID should be a global unique ID expressed in hexadecimal letters and 24 letters long, for example 0A11449F652C394A34DE042F. If an asynchronous request can be transmitted without errors inside the SAP system, the SAP Adapter will return a BizTalk envelope with an empty body as confirmation document. If there were any processing errors, the body will contain a fault descriptor element. The result of an asynchronous call can be checked inside the target system by using the ALE monitoring services (BD87) in a 4.6 SAP system. Please refer to your SAP system documentation for further information on ALE services.

Example

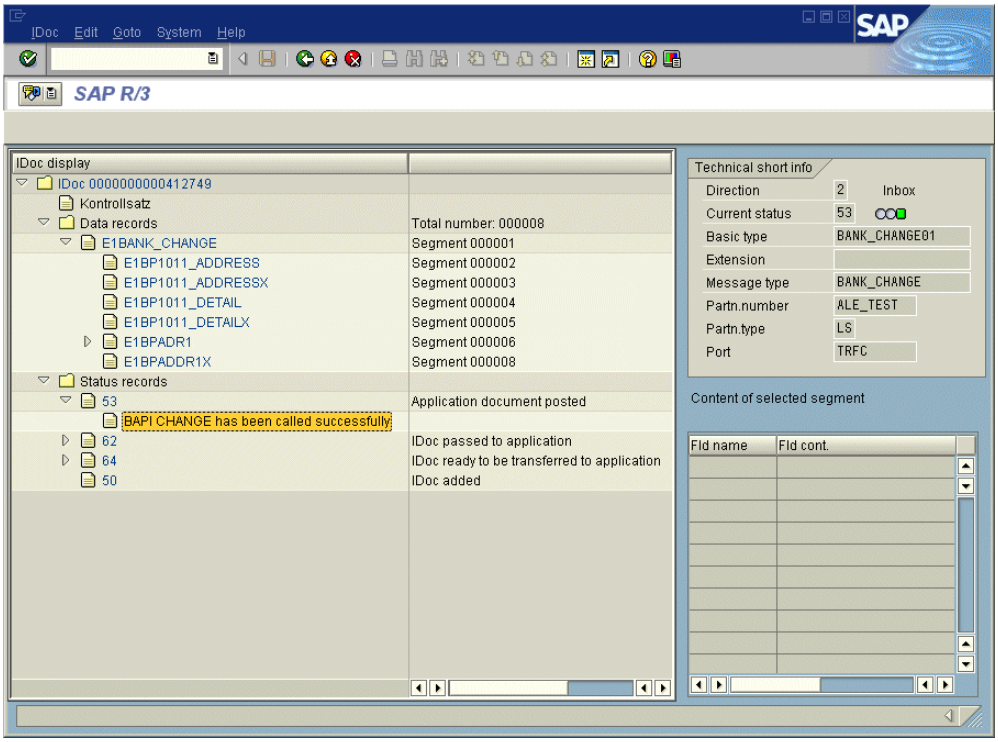
- Sending an asynchronous request to an SAP system by specifying a transaction ID



- Receiving an XML confirmation document



- Checking the correct application-level processing inside the target SAP system (BD87 in a 4.6 system or use the IDoc list WE05):



Routing IDocs Through the SAP Adapter

If you submit an IDoc-XML document in one of the following ways, it will automatically be passed to the routing listener.

Posting an IDoc-XML Document from an HTTP Client

Use the following guidelines to submit an IDoc-XML document via any HTTP client:

```
POST /invoke/pub.sap.transport.ALE/InboundProcess HTTP/1.1\r\n
Host: <hostname of client>\r\n
Content-Type: application/x-sap.idoc\r\n
Authorization: Basic <base64-encoded user:password>\r\n
Content-Length: <number of bytes in XML document>\r\n
\r\n
<?>xml version="1.0"?>

<MATMAS02>
...
</MATMAS02>
```

Note: The value for the content length has to be replaced by the actual length of content.

Tip! To simulate an HTTP client for testing purposes, you can use the service pub.client:http.

1 Using the Developer, navigate to the service pub.client:http.

2 Select **Test** ▸ **Run**.

3 Specify the URL of the IDoc-XML handler service. For example,
`http://localhost:5555/invoke/pub.sap.transport.ALE/InboundProcess`.

4 Specify **Post** as the method.

5 Add one entry in the field headers:

Name: `Content-type`
Value: `application/x-sap.idoc`

6 Copy a sample Idoc-XML document into the field **data** ▸ **string**

Posting an IDoc via FTP

Perform the following steps to FTP an IDoc document to the routing listener:

```
open <Integration Server host> <port of ftp listener>
username: Administrator
password: manage
cd ns/pub/sap/transport/ALE/InboundProcess
bin
put example.idocxml
```

The FTP client should check the return code to find out wether processing of the document was ok.

Note: It is important that the file name of the document ends with `.idocxml`.

Posting an IDoc in an E-mail Message

To send an IDoc document to the routing listener in an e-mail message, put the document into an attachment whose name ends with `.idocxml`. In the subject line of the e-mail specify pub.sap.transport.ALE:InboundProcess.

Posting an IDoc from a Web Browser

Note: To post an IDoc-XML document into the routing listener, you can submit it via a Web page or you can create an HTTP client that performs the post. This section describes how to post IDoc-XML documents from a web browser.

You can submit an IDoc-XML for any IDoc type to the routing listener from a Web browser using the following page:

```
http://<host>:<port>/WmSAP/submitIDocXML.html
```

This sample page can be used as a test tool. If you plan to submit IDocs from an HTML form, you can use this page as sample code.

Transmitting IDocs between Two Integration Servers

To send IDocs over the Internet, use the following procedure to establish the connection between two Integration Servers (for example: your own and one belonging to your Business Partner).

1 On the sending Integration Server, create a Remote Server that points to the receiving Integration Server. For more information on creating Remote Servers see *webMethods Integration Server Administrator's Guide*. If you are inside a firewall make sure that your HTTP proxy server is configured under Settings ▸ Proxy Servers.

2 On the sending Integration Server, create a new routing notification that has a new empty flow service assigned.

3 Set Transport to "IS".

4 Edit the service assigned to the routing notification and add a flow step invoking service pub.remote:invoke. Set the input values for this flow step as follows:

   a $alias: *Remote Server name of your trading partner*

   b $service: pub.sap.transport.IS:InboundProcess

   c $scope: select the default value

5 Edit the service assigned to the routing notification and drop all unnecessary parameters in a MAP located before the step invoking the pub.remote:invoke service. This reduces the amount of data sent over the network. If the receiving Integration Server is of release 4.6 or lower, you need to invoke another service before the final remote invoke step: pub.sap.idoc:iDocToTables. This service converts the IDoc into the format expected by older releases.

Tip! One more tip to increase performance: ask your communication partner to also drop all unnecessary parameters at the end of the routing notification which receives this document. This reduces the amount of data sent back in the response.

Mapping IDocs to Other Formats

If you need to use the information contained in an IDoc in documents of another format, you build a service that "maps" the information from fields in the IDoc to the variables used by the other application. For example, to transfer a purchase order from an ORDERS02 IDoc to an EDI system, you create a flow service that maps information from the IDoc fields to fields within the EDI system's purchase-order document. Then, you define a routing notification that triggers this flow service.

The following instructions will show you how to do this in detail:

1 "Creating an Empty Flow Service"

2 "Creating the Routing Notification"

3 "Creating a Document Type for Your IDoc"

4 "Transforming the IDoc to a Hierarchical Format"

5 "Mapping IDoc Information to Pipeline Variables"

6 "Testing the Mapping Service"

Creating an Empty Flow Service

Use the following procedure to create a flow service named app.idocs:mapOrders02 with Developer (this is the service that the routing listener will invoke based on the routing rule created in the next stage).

To create an empty flow service

On the File menu, click New and create an empty flow service. You will select this service when creating a routing notification.

Creating the Routing Notification

The first step in building an application that maps information from an IDoc is to create a routing notification that invokes the flow service that will perform the mapping.

When you create the routing notification, select the service you have previously created for mapping the IDoc, and as the transport, select "IS Service".

The following example shows the routing notification you would create to pass the IDoc to a service called app.idocs:mapOrders02.

Creating a Document Type for Your IDoc

Regardless of the way in which you pass the IDoc to the flow, you need to create a document describing the content and structure of the IDoc. There are three ways to create this document:

- You can generate it directly by receiving the IDoc metadata information form an SAP system you have an RFC connection to. See "Create an IDoc Document Type Using the DDIC" on page 120 for more information.
- You can generate it from the DTD for the IDoc. See "Generating an IDoc Document Type from a DTD" on page 121 for more information.
- You can generate it from a "sample" IDoc that you capture with the Developer at design time. See "Generating a Document Type from a Sample IDoc" on page 122 for more information.

The first two options produce the most complete document because they are based on the IDoc's metadata defined at the SAP system or a DTD, which describes all possible fields in the IDoc. Use one of the first two options whenever possible.

Use the third option only if a DTD is not available for the IDoc with which you are working. Because this option derives a document from a sample document, there is no assurance that the document will include all possible fields. Fields that are not used in the sample document will not appear in the document type.

Transforming the IDoc to a Hierarchical Format

To extract information from an IDoc, your flow service must transform the IDoc into a hierarchical structure and generate a *document*—a document type whose elements correspond to the fields in the IDoc. To do this, either use the iDocToDocument service (if the IDoc will be received by the SAP Adapterr via tRFC or via http POST with Content-Type: `application/x-sap.idoc`) or the xmlNodeToDocument service (if the IDoc will be received via http POST with Content-Type: `text/xml`).

To transform the IDoc to hierarchical format

1 Click  on the Flow Pane toolbar and select the pub.sap.idoc:iDocToDocument service or the pub.web:xmlNodeToDocument service. (If this service does not appear in the list, select Browse... to locate it.) Either service transforms the IDoc into a *document* variable, which contains the contents of the IDoc in a format that you can map to other pipeline variables.

2 Click  to save this flow service.

Mapping IDoc Information to Pipeline Variables

After transforming the IDoc into a *document*, you can map the fields of the IDoc to other variables in the pipeline (for example, to variables in a cXML or OAG document). To do this, you must add a MAP operation to the flow, insert document types for the IDoc and the target document, and then map fields in the IDoc to the appropriate variables in the other document.

The following procedure describes how to add the MAP operation to the service and configure the MAP operation to copy values from the IDoc to other variables in the pipeline.

To map the IDoc to variables in the pipeline

Perform the following steps to add a Document Type describing the content and structure of the IDoc.

1 In the Developer, select the **Flow** tab.

2 In the Flow Pane, select the **iDocToDocument / xmlNodeToDocument** operation.

3 Select the **Pipeline** tab and select the *document* variable under **Pipeline Out**.

4 Click  on the toolbar and select **Document**.

5 In the **Name** field, type the fully-qualified name of the document type you created in "Creating a Document Type for Your IDoc" on page 158, or select it in the **Folder** tree.

6 Click **OK**.

7 Type a name for this document and press ENTER. (You can give this Document any name you like, but we suggest that you identify the IDoc in the name.)

When you finish this step, **Pipeline Out** should contain a document that defines the structure of your IDoc. (Do not save the flow at this stage, because if you do, the yet unused document will be deleted again from the service...)

8 Take the following steps to map the *document* variable to the document you just created.

9 In **Service Out**, select the *document* variable.

10 In **Pipeline Out**, select the IDoc document and click **Map**, or map *document* to the IDoc document using 'drag and relate':

11 The Pipeline Editor will show a connecting line between *document* and the IDoc document you created in the previous procedure.

Perform the following steps to define the variable(s) to which you want to map information from the IDoc.

To define the variable(s) to which you want to map information from the IDoc

1 Select **Map** on the **Flow Pane** toolbar.

Note: Make sure the MAP operation appears immediately after the `iDocToDocument / xmlNodeToDocument` operation. If necessary, use the arrow buttons on the toolbar to move it to the correct position.

2 Select the **Pipeline** tab.

3 In **Pipeline Out**, select the bottom most variable, and then

4 Click  on the toolbar, and select the Document Reference.

5 In the Name field, type the fully-qualified name of the document definition you want to map the IDoc to, or select it in the Folder tree. Click OK.

6 Type a name for this Document and press ENTER.

When you finish this step, Pipeline Out should contain a document that defines the structure of your target document type. (Do not save the flow at this stage, because if you do, the yet unused document Reference will be deleted again from the service.)

7 Use Map to map fields from the IDoc Document in Pipeline In to the appropriate target variables in Pipeline Out. If you need additional information about this step, see the *webMethods Developer User's Guide*.

The following example shows a variable mapped from the ORDERS02 IDoc to a String variable in Document called PurchaseOrder.

Note: You can find a document type for the example purchase order shown above in sample.sap.records:PurchaseOrder.

Testing the Mapping Service

To test the flow service you created, use the SAPGui or the /WmSAP/submitIDocXML.html utility to submit your sample IDoc to the SAP Adapter. Make sure to specify the sender, receiver, and message type that you used for the routing rule that you created in "Configuring a Routing Notification" on page 132.

Check the results of the service to ensure that the IDoc is being mapped correctly.

If you want to debug your service in the Developer, you can use the savePipelineToFile and restorePipelineFromFile services to capture a submitted IDoc

1 Use the SAPGui or the /WmSAP/submitIDocXML.html utility to submit an IDoc to your flow service. When the service executes, the savePipelineToFile operation will make a copy of the pipeline (which will include your IDoc) and save it to file.

2 Delete the savePipelineToFile service and insert the restorePipelineFromFile service.

3 Select the Test ▸ Run command to execute the flow service. When it executes, the restorePipelineFromFile service will retrieve the copy of the pipeline containing your IDoc, which the remainder of the flow will operate on.

When you are finished testing, delete the restorePipelineFromFile service and save the finished flow.

Content-Based Routing and Mapping for IDocs

Occasionally, you must do some manipulation on IDocs before processing them in the SAP Adapter or before sending them to an SAP system. This will happen with customized IDocs that need specific processing. The two most common manipulations are *content based routing* and *mapping*. Both mechanisms can be performed on IDocs routed through the SAP Adapter. For IDocs going to an SAP system, only mapping manipulation applies.

Note: '*Content based routing*' enables you to specify routing parameters (sender, receiver and msgType) that override parameters in the header of the incoming IDocs. You determine such headers from the content of the IDoc itself. For example, you want to provide routing parameters like 'vendor' or 'sales organization' in a PO as information for the receiving system (instead of just the logical system's name).

'*Mapping*' is the mechanism that adds (or removes) data segments to (from) a document. Use this if you want to include the parameter 'sales organization' in the ORDERS IDoc.

A practical example for using the pre-routing mechanisms features is if you have a special Orders IDoc that contains its receiver and sender information in specific fields in the IDoc itself. It also contains extra fields that you want removed, or added, before you process the IDoc as an ORDER02 IDoc. You register special pre-routing services based on message types. You also can create a standard processing service to be executed for all message types and provides default routing information.

You can register both inbound and outbound manipulation services. The inbound services are executed for inbound IDocs received by the SAP Adapter via a RFC Listener or by invoking the pub.sap.transport.ALE:InboundProcess service (for example: over HTTP). The outbound services are executed before an IDoc is sent to an SAP system by pub.sap.transport.ALE:OutboundProcess.

To register an inbound or outbound mapping service

1. For the inbound case, create one or more services that implement the pub.sap.transport.ALE:aleRoutingInfo and/or pub.sap.transport.ALE:aleRoutingInfo_Default specification. For the outbound case, the services need to implement either the pub.sap.transport.ALE:aleMappingInfo and/or pub.sap.transport.ALE:aleMappingInfo_Default specification.
2. After having implemented a service, you still have to make it known to the IDoc transport. Go to Adapters ▸ SAP Adapter ▸ Routing/Mapping.
3. Follow the link Register New User Exit Service and then select one ALE Routing/Mapping Info * type.

4 Provide the full service name and the message type in the input field, and then click Register.

A routing/mapping service has to use the following specifications for proper behavior:

Specification	Description
pub.sap.transport.ALE:aleRoutingInfo	Specification that should be used for services which provide an inbound content based routing/mapping.
pub.sap.transport.ALE:aleRoutingInfo_Default	Specification that should be used for services which provide a default inbound routing/mapping
pub.sap.transport.ALE:aleMappingInfo	Specification that should be used for services which provide an outbound content based mapping.
pub.sap.transport.ALE:aleMappingInfo_Default	Specification that should be used for services which provide a default outbound mapping.

After you registered your routing services you can associate them with message types via the Administrator UI (Adapters > SAP Adapter > Routing/Mapping):

Important! In the SAP Routing/Mapping screen of the Administrator UI there are *two sections* to select services: a message type independent section (default handling section) and a message type dependent section.

A *default* handling service (inbound or outbound) is executed *for all message types*. If there are services selected in the message type dependent section, they will be executed for the corresponding message type *in addition to the default service*. If the special value $none is selected, all registered services are set to inactive.

Note: The services for content based routing have to be registered by yourself before they can be selected via the Administrator UI.

Important! If you create your own default inbound routing service, you must set the parameters sender, receiver and msgType. Those parameters are part of the specification. If you do not set them, you could receive an error message.

Note: The setting of the default services works the same way as the setting of regular services. You need to register a service before you can select it. If no "* Default" Routing/Mapping type has been registered, then you will not have a choice for selecting one at all, which is the case with the adapter out of the box. $none choice is only there if one or more services are registered.

For all incoming IDocs with an ORDERS message type, the sample.sap.idoc.Mappings:orders service will be executed. The service may alter the IDoc as well as the routing parameters for the IDoc. For IDocs going to the SAP system with ORDRSP message type, the sample.sap.idoc.Mappings:ordrsp service will be executed. This service can alter the IDoc by adding or removing fields.

Routing Arbitrary XML Documents Through the SAP Adapter

It is possible to use XML Inbound Process to forward arbitrary XML documents to the routing listener by using the Inbound Process mechanism of the XML transport.

For this purpose you can configure a so called inbound routing info service for your XML transport. This service should implement the specification pub.sap.transport.XML:xmlRoutingInfo, using the following parameters:

Input Parameters

This key	Must specify...
document	This field contains the XML document as it would look like after having been processed by pub.xml:xmlNodeToDocument. Your service should extract the routing data from your specific documents.
node	Alternatively, you can provide your document as a node as used in the pub.xml:xmlNodeToDocument service as an input field.

Return Values

This key	Must specify...
sender	Sender used for finding the matching routing notification.
receiver	Receiver used for finding the matching routing notification.
msgType	Message type used for finding the matching routing notification.
$tid (optional)	Transaction ID found in the document, if client wants to execute a transaction once and only once.
$action (optional)	One of the following transaction codes:

Code	Meaning
1	Execute (default)
4	Confirm

To configure the Inbound Process mechanism of the XML transport, perform the following steps:

1 Create an inbound routing info service for your XML transport. It extracts all required parameters so that the routing listener can handle the request.

2 After having implemented a service, you must make it known to the XML transport. Go to the Adapters ▸ SAP Adapter ▸ Routing/Mapping page.

3 Follow the link Register New User Exit Service and then select XML Routing Info.

4 Provide the full service name in the input field, and click Register:

Now you are ready to do routing using XML Inbound Process for arbitrary XML documents.

webMethods

CHAPTER 9

Transaction Handling

Managing Transactions and the Transaction Store

The SAP Adapter comes with an transaction manager that allows to monitor the transaction state for all messages associated with a transaction ID that uniquely identifies the transaction. The transaction ID can be up to 24 alpha numeric characters in length. The message itself can be stored into the transaction store. If the SAP Adapter receives a message without a transaction ID, it can create one for the message.

Use the Transactions screens to view and manage the transactions that the SAP Adapter records in its transaction store.

Here you can view transactions and their processing log as well as delete transactions that are no longer needed.

The Administrator should manually follow-up transactions in status Rolled back as follows:

Try to find and eliminate the reason of failure and then:

- Re-send the transaction from SM58, if it originally came from an SAP system
- Ask your partner to re-send the transaction, if it originally came in via FTP or HTTP.
- If the transaction was initiated by a local service, re-invoke the service.

Viewing Transactions in the Transaction Store

To view the transactions in the transaction store

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 In the SAP Adapter menu, click Transactions.

The SAP Adapter displays transactions in pages of 20, to avoid a timeout or out-of-memory of the web browser in case there is a large number of transactions in the store.

3 To view detail information for a transaction, click on the transaction ID in the TID column.

This screen allows you to view the details of each transaction, including the sender, receiver, message type, transaction ID, time when this transaction was first received by the SAP Adapter, current state, the time when this state was set, last error (if any), and an audit trail that shows state changes, error messages and the various steps that have processed the message.

The Current State field contains the transaction state. The information in this field corresponds to the different states of the tRFC protocol. If the message was not received via tRFC but via a different protocol (like http), the transaction store tries to imitate the tRFC status handling, but the meaning of the single states is slightly different in this case. The following describes the valid states of a tRFC transaction:

Status	Meaning (tRFC)	Meaning (other protocols)
Created	The sender has send a transaction ID, which was accepted by the SAP Adapter transaction manager. (No data sent yet.)	The sender has send a transaction ID and data, which was forwarded to the transaction store.
Rolledback	Execution of the transaction has failed. The sender may retry the transaction again at a later time.	Execution of the transaction has failed. The sender may retry the transaction again at a later time.
Committed	The sender has acknowledged, that he knows, that the transaction executed successfully.	Execution of the transaction on the SAP Adapter finished with a success message.
Confirmed	The sender has promised never to send this transaction again. So the TID may be deleted, as there is no need anymore to protect against duplicate processing of the same transaction.	This is used only if the external client invokes the service pub.sap.transport.*:InboundProcess again, with $action set 4, after the transaction has been executed successfully. In this case and if the transport supports it, the Confirm event has been forwarded to the final receiver, so that it is able to clean up its ARFCRSTATE table and remove the TID from it.

If the storing of message bodies has not been disabled on the Routing or RFC listener, it is also possible to view the pipeline or the message body of the transaction from this screen.

You can view the following:

- message body as XML
- message body as HTML (IDoc only)
- complete pipeline

Deleting Transactions from the Transaction Store

If the transaction store contains transactions that are no longer needed, you can delete them.

To delete transactions from the Transaction Store

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 In the SAP Adapter menu, click Transactions.

3 If you want to delete just a single transaction, click the icon in the Delete column for that transaction.

Automatic Cleanup of the Transaction Store

If the grows of the transaction store becomes a problem in your system, you ought to consider scheduling a periodic job that periodically purges the store of transactions, which are no longer needed. The following procedure can be used for this.

To purge certain transactions from the transaction store periodically

1 In the Developer, create a flow service that invokes the pub.sap.transaction:sweep service.

2 Pass the following parameters to the sweep service using Set Value.

For this parameter...	Specify...
state	The transaction state.
elapsedTime	The number of minutes that the transaction has been in status state
maxTrxCount	The maximum number of transactions to delete each time that pub.sap.transaction:sweep runs

3 In the Administrator UI go to Server ▸ Scheduler and schedule this flow service to run periodically.

Example

If you want to periodically purge all transactions which have been in the Confirmed state for longer than 30 minutes, you can create a service called app:purgeConfirmedTRX with those state and time parameters. You also specify how many transactions every invocation of the service will purge. This is so you can control the pace of your maintenance jobs

without putting to much load on the Server at one single time. It is also a good idea to schedule the job at times, when there is not much load on the Server.

Configuration Parameters for the Transaction Manager

The transaction store reacts to the following configuration parameters, which can be set in the file <*install_dir*>/config/server.cnf or per routing/RFC listener or on the notification level. Some of these influence the performance.

- On each RFC Listener and on the routing listener you can set field “Store message body” to “On” or “Off”.

  If this parameter is set to “Off”, the SAP Adapter will not save the message body of incoming transactions.

- On each RFC Listener and on the routing listener you can set field “Log transaction status” to “On” or “Off”.

  If this parameter is set to “Off”, the SAP Adapter will not maintain any transaction and status information in the transaction store.

- watt.sap.xtn.cacheFlushPeriod

  This parameter specifies, in minutes, how often the SAP Adapter will flush unsaved changes in the transaction cache to the file system. The default is 5 minutes. If a transaction in the cache has been modified (for example: new state, new audit log, etc.), it will be persisted to the transaction store after “cacheFlushPeriod” has expired and no further modifications have happened. After that it will remain in the cache until the “cacheTimeToLive” period has expired, or longer if it was modified again.

- watt.sap.xtn.cacheTimeToLive

  The transaction cache works both as a read and a write cache. After a transaction is read the first time, it is kept in the cache for a specific amount of time (cacheTimeToLive). If the transaction is not accessed during this time, it will be removed from the cache after this period has ended. This keeps the cache from growing indefinitely.

- On each ALE Notification and for the Routing Notification you can set field “Monitor IDocs” to “On” or “Off”.

  If this parameter is set to “On”, the SAP Adapter links the IDoc packet's TID with the DOCNUMs of the IDocs in that packet, so that later ALE IDoc Monitoring will be possible. See the section “Using the ALE Monitoring Features Via the SAP Adapter” on page 172 for more information.

Using the ALE Monitoring Features Via the SAP Adapter

Introduction

When an IDoc is sent from an SAP system to the SAP Adapter, the status of the IDoc (as displayed in WE02) will always be "03, Data passed to port OK" (passed to the tRFC queue of the SAP system). However, this status does not provide any information regarding whether the IDoc could be processed successfully by the SAP Adapter and who the final recipient was. To get this kind of information, the SAP Adapter offers three options: IDoc Trace, ALEAUD IDoc and SYSTAT IDoc.

The IDoc Trace feature should be used if you sporadically want to look up the status of certain IDocs. The lookup is done manually and only for a specific selection of IDocs. If you want *automatic* status update, you should use one of the other two options described below.

Status update via ALEAUD IDoc can be used if the final recipient is again another SAP system. The most common case will be the connection of two SAP systems via the Internet like this:

> SAP system (sender) -> Integration Server 1 ->Internet(http) -> Integration Server 2 -> SAP system (receiver).

In all other cases status update via SYSTAT IDoc should be used. Integration Server acts here like an EDI Subsystem and returns status information and information about the final receiver (receiving e-mail address, Web Server URL, FTP Server, third party system, etc.) to the sender.

These three options are now described in detail:

IDoc Trace

Overview

The ALE Monitor allows you to make inquiries about the processing status of an IDoc in the receiving system actively from within the sending system. In SAP systems of release < 4.6C this can be done with the transaction BDM2 (*Cross System IDoc Reporting*) and from release 4.6C on with transaction BD87 (*Status Monitor for ALE Messages*). As inputs you can specify certain selection criteria, like Document Number, Message Type, LS of receiver and date ranges. The system then makes a synchronous RFC call (IDOC_DATE_TIME_GET) to the LS that received the IDocs, with a list of DOCNUMs corresponding to the selected IDocs. (From transaction BD87 you have to hit the toolbar button "Trace IDocs" after the selection of the IDocs.) This function call returns for each IDoc the DOCNUM, under which it was saved in the receiving system, the receiving time and the current processing status in the receiving system.

Prerequisites

- The IDoc Trace functionality only works for IDocs exchanged between partners of Partner Type LS (Logical System).
- In order to be able to make this synchronous RFC, the partner profile of the LS, which represents the Integration Server, must contain an outbound parameter with message type SYNCH.
- Also the Function Call to an LS of type "tRFC Port", like the Integration Server, is only executed if the SAP system is of release 4.6D or higher. For older releases you have to apply the following hot packages:

R/3 Release	Hot Package
3.1I	SAPKH31I67
4.0B	SAPKH40B56
4.5B	SAPKH45B35
4.6B	SAPKB46B22 and SAPKH46B22
4.6C	SAPKB46C11 and SAPKH46C11

Prepare the SAP Adapter for IDoc Trace

To set up the SAP Adapter for IDoc tracing proceed along the following steps:

1 For your routing notification or ALE adapter notification you want to monitor IDocs, set the value for field "Monitor IDocs" to "On" using the Developer.

2 For each SAP system from which you want to receive and trace IDocs, create an RFC adapter notification for IDOC_DATE_TIME_GET and assign service pub.sap.monitor.idoc:trace.

Note: Depending on how the SAP Adapter processes the IDoc it receives, different information will be returned. By default, the creation date and time at the receiving system and the status will be returned. If the IDoc will not be forwarded to another SAP system, the SAP Adapter is the receiving system.

However, if the IDoc is forwarded to an SAP system (either by invoking a routing notification with outbound transport set to ALE or by invoking pub.sap.transport.ALE:OutboundProcess or pub.sap.client:sendIDoc directly or from a remote Integration Server) the SAP Adapter invokes IDOC_DATE_TIME_GET directly against the receiving SAP system. This action returns additional information, like the IDoc number, from the receiving SAP system. All remote Integration Servers that receive IDocs via a remote invocation of pub.sap.transport.ALE:InboundProcess should therefore also support IDoc Tracing.

Status Update Via ALEAUD IDoc

Preparation

If the final receiver is an SAP system, the IDoc ALEAUD can be used to report status information from the receiver back to the sender. To setup this scenario, the following settings have to be created in the distribution models of the participating Systems:

(MATMAS is used in this example)

- Sending SAP System: For the LS that represents the receiver, set up a partner profile, that has MATMAS as an outbound parameter and ALEAUD with process code AUD1 as an inbound parameter.
- Receiving SAP System: For the LS that represents the sender, set up a partner profile with an outbound parameter ALEAUD and an inbound parameter MATMAS. Also in the distribution model the following model view has to be created:

  Sender: T90CLNT090 (or LS, which received the IDoc, if different)

  Receiver: LS of sender

  Message type: ALEAUD

Here add a Filter with value "MATMAS".

Next you have to schedule a job, which executes a variant of the report RBDSTATE. The variant has to include the LS of the sender as selection parameter. This should send out an ALEAUD IDoc to the Integration Server.

Further Setup in the SAP Adapter

To set up the SAP Adapter for status update via ALEAUD proceed along the following steps

1. For the routing notification or ALE adapter notification you want to use to monitor IDocs, set the value for the Monitor IDocs field to On using the Developer.
2. After all this has been set up, the ALEAUD IDoc can be routed as usual like a "normal" IDoc. Here –and in all other cases, where sender/receiver information in the IDoc Control Header has been changed by a mapping—only one thing has to be taken into account: the last Integration Server, which pushes the ALEAUD into the original sending system, has to set the values of EDI_DC40-SNDPRN and EDI_DC40-RCVPRN to the inverse of their original values.

For example: If your original IDoc went out with a header segment such as:

```
SENDER              Receiver
LS T90CLNT090       LS XYZCLNT123
```

Then the ALEAUD has to be pushed in with this header:

```
EDI_DC40-SNDPRN = XYZCLNT123
EDI_DC40-RCVPRN = T90CLNT090
```

This can be achieved by adding the service pub.sap.monitor.aleaud01:swapSenderReceiverLS as preprocessing service to the routing notification service.

Status Update Via SYSTAT IDoc

Overview

In this case the Integration Server can be considered as a kind of EDI Subsystem. If the SYSTAT feature is enabled, it automatically sends a customary EDI Subsystem Status and some additional information for each IDoc it received back to the original sender. You can decide for each routing notification or ALE adapter notification whether SYSTAT information should be reported to the sender or not. The following information is then reported in addition to the status:

- Program that set the status (SAP Adapter)
- routing notification or ALE adapter notification that processed the IDoc
- The key (TID) under which the IDoc can be found in the SAP Adapter
- Date and time of status change
- In case of an error status: explicit error message
- In case of a success status: explicit information about the final receiver

The SAP Adapter sends the following status:

- If everything went ok the first time: "`06 Translation OK`" and "`12 Dispatch OK`"
- If an error has occurred the first time: "`11 Error during dispatch`"
- If everything went ok at a later retrial: "`13 Retransmission OK`"
- If there was still an error at a later retrial: "`23 Error during retransmission`"
- If no information on the IDoc could be determined: "`04 Error within control information of EDI subsystem`"

Setup the participating SAP Systems

In each SAP system that is to receive SYSTAT IDocs from the Integration Server, you need to configure the following settings:

In SAP systems of Release 3.1H - 3.1I you need to perform the following additional step, before continuing with the general setup:

Go to WE42 (Process codes, inbound) and open the two tree branches Inbound with ALE service ▸ Processing by task and Inbound without ALE service ▸ Processing by task. You need to reassign the process code STA1 from without ALE service to with ALE service, if this has not already been done. To do this, proceed as follows:

1 Mark the node Inbound without ALE service ▸ Processing by task ▸ STA1 Status record from IDoc with the cursor and then press the Reassign button (F6).

2 Confirm the popup

3 Double click on the node Inbound with ALE service _ Processing by task

4 In the following screen press the Save button (F11).

General setup:

1 Open SALE (Distribution (ALE)) and select Basis configuration ▸ Set up logical system ▸ Maintain logical systems to add a new logical system for the Integration Server if you do not have one yet. If you name it `BUSCON`, the Integration Server will recognize it automatically; otherwise, you will have to make the name known to the Integration Server as described later.

2 In WE20 (Partner Profiles), create a partner profile with Partn.number = the name you chose in step 1 (`BUSCON`) and Partn.type = LS. After you saved it, add an Inbound Parameter to it as follows:

   Message type = `STATUS`

   Process code = `STA1`

Prepare the Integration Server for Automatic SYSTAT IDoc

In the Integration Server, you have to setup these elements to enable automatic status update via SYSTAT:

1 If you chose a different name from `BUSCON` for the logical system above, shutdown the Integration Server and add the following parameter to ...\IntegrationServer\config\server.cnf:

```
watt.sap.systat01.partnerNumber=<NameOfLogicalSystem>.
```

Start the Integration Server again.

With `BUSCON` you can omit this step.

2 To mark a routing notification or ALE adapter notification for automatic status update, edit the notification and set the value for field “Monitor IDocs” to “On”.

3 Go to **Server ▸ Scheduler ▸ Create a scheduled task** and schedule the service pub.sap.monitor.systat01:report.

Here you specify time intervals for the Service. You should try to schedule it for times when you know that there is little load on the system. On the other hand, in order to prevent the resulting SYSTAT IDocs from getting too big, you should schedule it often enough, so that at most around 2000 IDocs have been received by the SAP Adapter since the last run of the report service.

This service collects all the necessary information and error/success messages for those IDocs, which the notifications chosen in step 2 have processed since the last run. For each SAP system that has sent any such IDocs to the SAP Adapter, it creates and submits one SYSTAT IDoc containing all this information.

webMethods

CHAPTER 10

Logging and Monitoring

Logging

In the development and test phase it is sometimes helpful to view RFC traces if there are problems with the function modules you are trying to call. In order to avoid the need to access the file system of the machine on which the Integration Server is running, you can view and delete RFC trace files and SAP log files from the Administrator UI.

All SAP log files, as well as the RFC trace files are stored in the packages/WmSAP/logs directory. As part of the SAP Adapter installation, instructions were provided to set the RFC_TRACE_DIR environment variable to point to that directory. (See the *webMethods SAP Adapter Installation Guide* for information about setting the directory for trace files.)

If the RFC_TRACE_DIR environment variable was not set during adapter installation, the SAP Adapter trace files will be stored in the root directory of the Integration Server. You will not be able to view the trace files in the Administrator UI.

Viewing and Deleting RFC Trace Files and SAP Log Files

To view and delete RFC trace files and SAP log files

1 In the Administrator UI, browse to **Adapters** ▸ **SAP Adapter** ▸ **Logging**.

2 The Logging screen appears. This screen lists the RFC trace files and SAP log files, and shows each file's size and creation date.

- RFC trace files are split into several parts, each of them containing one trace entry along with the date and time it was created. To view the log entries for a specific Trace File, click the Trace File name.
- Large SAP log files are split into several parts of about 500K. To view the log entries for a specific SAP log File, click the SAP Log File name.

  In this way it is avoided that the browser has to load too large documents, which would take a long time or even cause a time out. Using the **Back** button of your browser, you can return from the display of one part to the list of parts for that file. If you try to delete a file that is still open (for example, because the Server is still writing to it), you will receive a message, and the file will not be deleted.

3 You can delete files that you no longer need:

- To delete an individual trace file or SAP log file, click the  icon for the file to delete.
- To quickly delete all trace files or all SAP log files, click **Delete All**.

Monitoring SAP Adapter Performance

Performance Output Information in the SAP Log File

From debug level 7 or higher you can view performance output information for the SAP Adapter in the SAP log file. The corresponding terms are described below:

	The term...	Specifies...
1	Time for marshalling (ms)	Time needed for transferring the data from IData (pipeline representation) to JCo Objects.
2	Time for unmarshalling (ms)	Time needed for transferring the data from JCo Objects to IData (pipeline representation).
3	Time for preparing (ms)	Time needed for preparing execution of a function module (outbound calls) or invocation of a service. This includes repository queries for the data structures and the function interface, opening connections to the involved systems, etc.
4	Time for rfc calls (ms)	Time needed in the JCo layer for outbound calls.
5	Time for SAP Adapter service calls (ms)	Time needed for the invocation of a service to handle an inbound request in the SAP Adapter after preparing/marshalling and before unmarshalling the data. (Listener performance data).
6	Total time for function calls (ms)	Total time needed for handling inbound/outbound calls in SAP Adapter layer.

Component Response Time Measurement

With this kind of monitoring, an application gets split into several components and the response time gets measured per component.You will find this information on Adapters ▸ SAP Adapter ▸ Monitoring:

At the bottom of this screen, you will find the following monitoring information:

Monitoring information

The term...	Specifies...
Requests total	Number of requests executed so far.
Request rate	Requests per second since monitoring was enabled.
Requests ok	Number of successful requests.
Requests with errors	Number of requests with errors.

The term...	Specifies...
Components total	Number of components used in requests.
Components per request	Average number of components per request.
Total time	Execution time for all requests in milliseconds.
Average time	Average execution time for a request in milliseconds.
Query Requests	You can get the TOP 100 requests (the requests that needed the longest execution times.) To restrict those entries, you can specify the Info for and max. listed Requests inputs before pushing the Show Button.
Info for ...	Enter a wildcard-like pattern for the request names, for which you want the information. You can use only exact patterns and patterns with a single '*' at the end, such as 'client.*'. You cannot make entries with the format '*rfc*'.
max. listed Requests	Limits the number of requests to the given number. After pushing the button, the Requests Overview is displayed.
Query Components	You can get accumulated information about all components, that have been used in Request. To restrict those entries you can specify the Info for input before pushing the Show Button:
Info for ...	Enter a wildcard-like pattern for the Component names, for which you want the information. You can use only exact patterns and patterns with a single '*' at the end, such as 'client.*' You cannot make entries with the format '*rfc*'. After pushing the button, the Component Overview is displayed.

webMethods

CHAPTER 11

Coding Client Applications and Services

Overview

The SAP Adapter has several APIs that you can use in your client applications and services. This chapter shows how to use the built-in services API to invoke SAP RFCs, send IDocs to an SAP system and to receive an IDoc from an SAP system. For descriptions of the available built-in services, refer to Appendix D, "Built-in Services".

For more information about building the Java services, see the online API documentation installed with the Integration Server at:
webMethods_directory\IntegrationServer\doc\api\Java\index.html

This chapter also shows how to use the webMethods SAP Adapter IDoc Java API to construct an IDoc. For information, refer to *webMethods_directory*\IntegrationServer\packages\WmSAP\pub\doc\api\index.html.

Invoking RFCs from the SAP Adapter

Calling Public SAP Adapter Services from Java Services

This section shows an example of how to invoke an RFC from a Java service. The sample app/BAPI.java that is shown below logs on to an SAP system and invokes a BAPI.

For more information about developing services, see the *webMethods Developer User's Guide* and the online API documentation.

To call a public SAP Adapter service from Java

1 Start the Developer if it is not already running.

2 Create a Java service named app.BAPI:callService and enter the following source:

```
IDataCursor idc=pipeline.getCursor();
IDataUtil.put(idc, "serverName", "CER");
IDataUtil.put(idc, "COMPANYID", "000001");
IDataUtil.put(idc, "$rfcname", "BAPI_COMPANY_GETDETAIL");
try
{
   Service.doInvoke("pub.sap.client", "invoke", pipeline);
}
catch (Exception e)
{
   throw new ServiceException(e);
}
finally
{
   idc.destroy();
}
```

This service will invoke BAPI_COMPANY_GETDETAIL with a company ID of 000001 and return the results. The logon to the SAP system is done automatically using the connection information for the RFC connection alias.

Note: Change the RFC connection alias (field serverName of the public SAP Adapter service) to your own alias.

Receiving IDocs from an SAP System

To receive an IDoc from an SAP system and pass it to a service, you can setup an synchronous ALE adapter notification or a routing notification. In either case, you need to assign the service you would like to invoke to the notification. For instructions on establishing routing notifications, refer to "Routing Notifications" on page 132. When choosing the transport for the routing notification, select the IS transport. For instructions on establishing asynchronous ALE listener notifications, refer to "Creating a Synchronous RFC Adapter Notification" on page 108.

Note: With Release 6.5 of the SAP Adapter, you must use the SAP Java IDoc Class Library to process IDocs.

Accessing and Modifying Fields in IDocs

The following code sample shows how to use the SAP Java IDoc Class Library to convert an IDoc to a structured document with direct access to each field. This allows you to modify an IDoc's contents on the fly. For example, if you want to customize incoming IDocs based on local data format, you can do so.

```
IDataCursor idc=pipeline.getCursor();
IDoc.DocumentList iDocList=null;
if (idc.first("iDocList"))
iDocList=(IDoc.DocumentList)idc.getValue();
```

under Shared ▸ Imports, include the lines:

```
com.wm.adapter.sap.idoc.*
com.sap.mw.idoc.IDoc
```

Note: To resolve the included Java classes, you must include the WmSAP package in the dependency section of your application package.

For more information about using the SAP Java IDoc Class Library, refer to "Constructing an IDoc with the SAP Java IDoc Class Library" on page 186

Converting an IDoc to XML

The following code sample shows how to convert an IDoc to XML.

```
String xmlData = idocList.toXML();
```

Constructing an IDoc with the SAP Java IDoc Class Library

The following sample shows how to construct a new MATMAS IDoc.

Important! Remember to include `com.wm.adapter.sap.idoc.*` and `com.sap.mw.idoc.IDoc` in Shared ▸ Imports.

Note: The first three lines are generated by the Developer.

```
public final static void createIDoc (IData pipeline)
throws ServiceException
{
//create a new and empty MATMAS02 document
IDoc.DocumentList iDocList = new IDataDocumentList();
IDoc.Document doc=iDocList.addDocument("MATMAS02");

//set the appropriate control header data
doc.setIDocNumber("0000047112211178");
doc.setClient("000");
doc.setDirection("1");
doc.setRecipientPartnerType("LS");
doc.setMessageType("MATMAS");
doc.setRecipientPartnerType("LS");
doc.setRecipientPartnerNumber("TSTCLNT000");
doc.setSenderPort("SAPBCIDOC");
doc.setSenderPartnerType("LS");
doc.setSenderPartnerNumber("SBCCLNT000");
doc.setCreationDate("20030511");
doc.setCreationTime("103051");
doc.setSerialization("20030511103051");

//get the root segment from the document
//The root segment does not contain any fields or data. It is only
//used as the standard parent segment and won't be transmitted when
//the document is sent to an SAP system.
IDoc.Segment segment = doc.getRootSegment();

//create and add a new and empty child segment of type E1MARAM
//and fill the segment data
segment = segment.addChild("E1MARAM");
segment.setField("MSGFN", "005");
segment.setField("MATNR", "BOXCOOKIES");
```

```
segment.setField("ERSDA", "20030303");
segment.setField("ERNAM", "TOPSI");
segment.setField("PSTAT", "KBG");
segment.setField("MTART", "FERT");
segment.setField("MBRSH", "L");
segment.setField("MATKL", "G1113");
segment.setField("MEINS", "PCE");
segment.setField("BLANZ", "000");
segment.setField("BRGEW", "0.550");
segment.setField("NTGEW", "0.000");
segment.setField("GEWEI", "KGM");
segment.setField("VPSTA", "KBG");

//create and add a new and empty child segment of type E1MAKTM
//and fill the segment data
segment = segment.addChild("E1MAKTM");
segment.setField("MSGFN", "005");
segment.setField("SPRAS", "D");
segment.setField("MAKTX", "Schachtel mit Keksen");
segment.setField("SPRAS_ISO", "DE");

//create and add a new and empty sibling segment of type E1MAKTM (same
//type) and fill the segment data
segment = segment.addSibling();
segment.setField("MSGFN", "005");
segment.setField("SPRAS", "E");
segment.setField("MAKTX", "Box of cookies");
segment.setField("SPRAS_ISO", "EN");

//create and add a new and empty sibling segment of type E1MARCM
//and fill the segment data
segment = segment.addSibling("E1MARCM");
segment.setField("MSGFN", "005");
segment.setField("WERKS", "0001");
segment.setField("PSTAT", "BG");
segment.setField("PLIFZ", "0");
segment.setField("WEBAZ", "0");
segment.setField("PERKZ", "M");
segment.setField("AUSSS", "0.00");
segment.setField("BESKZ", "E");
segment.setField("AUTRU", "X");

//create and add a new and empty sibling segment of type E1MBEWM
//and fill the segment data
segment = segment.addSibling("E1MBEWM");
segment.setField("MSGFN", "005");
segment.setField("BWKEY", "0001");
segment.setField("VPRSV", "S");
segment.setField("VERPR", "0.00");
segment.setField("STPRS", "15.50");
segment.setField("PEINH", "1");
```

```
segment.setField("BKLAS", "7920");
segment.setField("VJVPR", "S");
segment.setField("VJVER", "0.00");
segment.setField("VJSTP", "15.50");
segment.setField("LFGJA", "2002");
segment.setField("LFMON", "08");
segment.setField("PSTAT", "BG");
segment.setField("KALN1", "000100126602");
segment.setField("KALNR", "000100126603");
segment.setField("EKALR", "X");
segment.setField("VPLPR", "0.00");
segment.setField("VJBKL", "7920");
segment.setField("VJPEI", "1");
segment.setField("BWPEI", "0");

IDataCursor idc=pipeline.getCursor();
IDataUtil.put(idc, "iDocList", iDocList);
idc.destroy();
}
```

Sending IDocs to an SAP System

The basic steps to send an outbound IDoc from either a client or a service are:

1 Create a transaction ID by invoking the pub.sap.client:createTID service.

2 Send the IDoc to the SAP system by invoking the pub.sap.client:sendIDoc service, passing in the `com.sap.mw.idoc.IDoc.DocumentList` object (iDocList).

If an error occurs, repeat the pub.sap.client:sendIDoc service invocation with the same transaction ID. The SAP system guarantees that the transaction will execute only once.

3 After pub.sap.client:sendIDoc returns successfully, invoke pub.sap.client:confirmTID.

Transaction Features for HTTP and SAP-XML

When executing HTTP Posts using IDoc-XML and XRFC with transactions, you should make sure that your client correctly handles such calls. It should have transaction management that supports at least the following features:

- Transaction ID generation
- Resending documents with same transaction ID in error cases.
- After a successful execution of a transaction, a document cannot be sent again.

To transfer the transaction ID to the SAP Adapter, you add an extra header to the HTTP POST, x-tid, that contains the transaction ID.

Example: HTTP POST of an IDoc

```
POST /invoke/pub.sap.transport.ALE/InboundProcess HTTP/1.0
X-tid: 9B38FA81133A38B518A10036
Content-type: application/x-sap.idoc
Content-Length: 4242
  <?xml version="1.0" encoding="iso-8859-1"?>
  <MATMAS02>
    <IDOC BEGIN="1">
      <EDI_DC SEGMENT="1">
        <TABNAM>EDI_DC</TABNAM>
        <MANDT>000</MANDT>
        <DOCNUM>0000047112211178</DOCNUM>
        <DOCREL/>
        <STATUS/>
        <DOCTYP>MATMAS02</DOCTYP>
        <DIRECT>1</DIRECT>
...
```

For XRFC documents, you can also put the transaction ID in the header of the document:

```
<?xml version="1.0"?>
  <sap:Envelope xmlns:sap="urn:sap-com:document:sap" version="1.0">
    <sap:Header xmlns:rfcprop="urn:sap-com:document:sap:rfc:properties">
      <rfcprop:Transaction>0A1104DC080C3C60F9E106A1</rfcprop:Transaction>
    </sap:Header>
    <sap:Body>
      <rfc:SBC_TEST_SAVE xmlns:rfc="urn:sap-
          com:document:sap:rfc:functions">
        <INPUT>Have fun!</INPUT>
      </rfc:SBC_TEST_SAVE>
    </sap:Body>
  </sap:Envelope>
```

Calling a BAPI Synchronously from an SAP System

This example demonstrates how to send the BAPI CompanyCode.GetDetail from an SAP system to the Web via XML.

As the BAPI will be handled by a routing notification, you need to specify some information to enable correct routing. For this purpose, you should use the parameter SBCHEADER when calling the RFC function module from the SAP system.

To set up for the example

1 Set up an RFC Listener for the SAP system on your SAP Adapter as described in "Listeners" on page 87.

2 Set up a corresponding RFC destination on your SAP system for your RFC Listener using the transaction SM59 as described in "Creating an RFC Destination on an SAP System" on page 84.

3 Create a program Z_BAPI_RFC_DEMO in the SAP system using the transaction SE38 You can enter the following source code:

```
REPORT Z_BAPI_RFC_DEMO .
*variable companyCodes will take the application result
data companyCodes like BAPI0002_1 occurs 1 with header line.
*variable header gives SAP Adapter instructions for routing
data header like SBCCALLENV occurs 1 with header line.
*variable returnCode takes the processing information after the
*synchronous call
data returnCode type BAPIRETURN.

*Fill the SAP Adapter routing instructions

header-name = 'sender'.
header-value = 'CERCLNT800'.
append header.
header-name = 'receiver'.
header-value = 'CERCLNT750'.
append header.

*Execute the RFC with destination ISCER
CALL FUNCTION 'BAPI_COMPANYCODE_GETLIST'
*->ISCER should be maintained in SM59 as alias to SAP Adapter
  DESTINATION 'ISCER'
IMPORTING
  RETURN = returnCode
TABLES
  COMPANYCODE_LIST = companyCodes
  SBCHEADER = header[]
          .
*process the BAPI-Return parameter
if not returnCode is initial.
  write: / 'BAPI return parameter:'.
  write: / ' Code: ' , returnCode-CODE.
  write: / ' Message: ' , returnCode-MESSAGE.
endif.
*display the application result
loop at companyCodes.
  write: / companyCodes-COMP_CODE, ' ', companyCodes-COMP_NAME.
endloop.
```

Before executing this report, you should specify a routing notification to the XML transport. You can send the XML message to any server that is able to process the HTTP Post message and send back a correct response. In this example, the call will be sent back to the locally installed Integration Server to demonstrate its inbound XML processing capabilities.

To prepare a routing notification for the BAPI call

1 Open the Developer and select New ▸ Adapter Notification. Click Next.

2 Select SAP Adapter from the list of available adapter types. Click Next.

3 From the list of available templates, select Routing Notification. Click Next.

4 Select the routing listener wm.sap.internal.ls:routingListener. Click Next.

5 Enter a name and select a folder where the routing notification should be stored.

6 Select any service that should be invoked by this notification. Click Next and then Finish.

Note: The service you selected in step 6 will only be used if the outbound transport that is chosen in your routing notification is "IS". For all other outbound transports, this selection will be overridden by a transport specific service.

7 In the input fields at the top of the page, enter the following data:

   a Sender: `CERCLNT800`.

   b Receiver: `CERCLNT750`.

   c msgType: `BAPI_COMPANYCODE_GETLIST.`

   d Select XML in the Transport field.

8 Specify a URL for the target system. To test the functionality using your local Integration Server's Inbound processing, enter: `http://localhost:5555/invoke/pub.sap.transport.BAPI/InboundProcess` (note that in some cases you have to adjust the port 5555 to your current settings).

   a Select dialect bXML for the xmlType parameter.

   b Select Yes for the useBAPI parameter.

   c Enter "CompanyCode" for the objectName parameter.

   d Enter "GetList" for the bapiName.

   e Click the Save button.

To handle the inbound XML message, which will now be sent back to the local Integration Server, you have to specify a second routing notification for the BAPI:

To specify a second routing notification for the BAPI

1 In the input field at the end of the page, enter the following data:

   a Sender: `CERCLNT800`.

   b Receiver: `CERCLNT750`.

   c msgType: `CompanyCode.GetList`.

2 Select BAPI in the Transport field.

3 Select the SAP system to which the message should be routed from the drop down list.

When executing the program in the SAP system, the SAP Adapter sends the HTTP Request to the remote host, which looks similar to the following:

```
POST /invoke/pub.sap.transport.BAPI/InboundProcess HTTP/1.0
User-Agent: Mozilla/4.0 [en] (WinNT; I)
Accept: image/gif, */*
Host: localhost:90
Content-type: application/x-sap.busdoc
Cookie: ssnid=553891555519
Content-length: 817

<?xml version="1.0" encoding="iso-8859-1"?>
  <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
    <header>
      <delivery>
        <message>
          <messageID>0A125F1315B3D24B0000001E</messageID>
          <sent>2000-06-20T09:58:00</sent>
        </message>
        <to>
          <address>urn:sap-com:logical-system:CERCLNT750</address>
        </to>
        <from>
          <address>urn:sap-com:logical-system:CERCLNT800</address>
        </from>
      </delivery>
    </header>
```

```
    <body>
      <doc:CompanyCode.GetList xmlns:doc="urn:sapcom:
        document:sap:business" xmlns="">
        <CompanyCodeList>
          <item>
            <COMP_CODE></COMP_CODE>
            <COMP_NAME></COMP_NAME>
          </item>
        </CompanyCodeList>
      </doc:CompanyCode.GetList>
    </body>
  </biztalk_1>
```

When using the Integration Server as the remote partner as described above, the BAPI will be executed in the selected system, the exporting and changing parameters will be put in an XML response document and sent back to the calling SAP system.

The report will then display the parameters received from the remote system, which in this case is a list of company codes.

Calling a BAPI Asynchronously from an SAP System

This example demonstrates how to call a BAPI asynchronously from an SAP system over the Web. For this purpose, you have to create an ABAP program inside the SAP system. This program calls the proxy function module to generate an ALE message for a BAPI. After performing a COMMIT WORK command, the SAP system will generate an IDoc through the ALE service layer and send it to the SAP Adapter. There the IDoc will be transformed into the original BAPI representation and sent as XML using HTTP to any remote web system.

Note: While BAPI function module names usually start with the prefix `BAPI_`, the corresponding ALE proxies are usually named equally with the prefix `ALE_`

To run an example, you should set up your SAP system for ALE processing as described in the SAP system documentation and in this guide. When defining the partner profile for the outgoing IDoc in the SAP system, ensure that the option Transfer IDoc immediately is selected because the BAPI conversion tool on the SAP Adapter does not support IDoc packages.

Create the following report in your SAP system to test the BAPI conversion with the BAPI Bank.Create (available as of SAP release 4.6C).

```
REPORT Z_BAPI_ALE_DEMO .
parameters receiver like BDI_LOGSYS-LOGSYS.
parameters country like BAPI1011_KEY-BANK_CTRY default 'DE'.
parameters bankkey like BAPI1011_KEY-BANK_KEY default '34981255'.

data: receivers like BDI_LOGSYS occurs 0 with header line,
      address like BAPI1011_ADDRESS.
```

```
*prepare the distribution information
move receiver to receivers-LOGSYS.
append receivers.

*prepare the BANK addres parameter
address-BANK_NAME = 'Demo Bank'.
address-REGION = 'BW'.
address-STREET = 'Neurottstr. 16'.
address-CITY = 'Walldorf'.
address-SWIFT_CODE = 'ABCDDE12'.
address-BANK_GROUP = 'SB'.
address-BANK_NO = '12345678'.
address-ADDR_NO = '123'.

*send data to ALE
CALL FUNCTION 'ALE_BANK_CREATE'
  EXPORTING
    BANKCTRY = country
    BANKKEY = bankkey
    BANKADDRESS = address
  TABLES
    RECEIVERS = receivers
  EXCEPTIONS
    ERROR_CREATING_IDOCS = 1
    OTHERS = 2
        .
IF SY-SUBRC <> 0.
  write: / 'IDoc could not be created'.
ELSE.
  write: / 'IDoc successfully created'.
ENDIF.

*trigger processing
commit work and wait.
```

To create similar routing notifications

You can create routing notifications similar to those described in the synchronous example.

1 In the input field, enter the following data:

   a Sender: enter the logical system of the sending system.

   b Receiver: enter the logical system of the target system.

   c msgType: BANK_CREATE.

2 In the transport field, select XML.

3 Specify an URL for the target system. To test the functionality using your local Integration Server's Inbound processing, enter:

`http://localhost:5555/invoke/pub.sap.transport.BAPI/InboundProcess`

(note that in some cases you have to adjust the port 5555 to your current settings)

4 Select dialect BizTalk for the xmlType parameter.

5 Select Yes for the useBAPI parameter.

6 Enter "bank" for the objectName parameter.

7 Enter "Create" as the BAPI.

To handle the Inbound XML Message

To handle the inbound XML message, which will now be sent back to the local Integration Server, you have to specify a second routing notification for the BAPI:

1 In the input field, enter the following data:

a Sender: enter the logical system of the sending system.

b Receiver: enter the logical system of the target system.

c Message-type: `Bank.Create`.

2 In the field transport, select BAPI.

3 Select the SAP system to which the message should be routed from the drop down list.

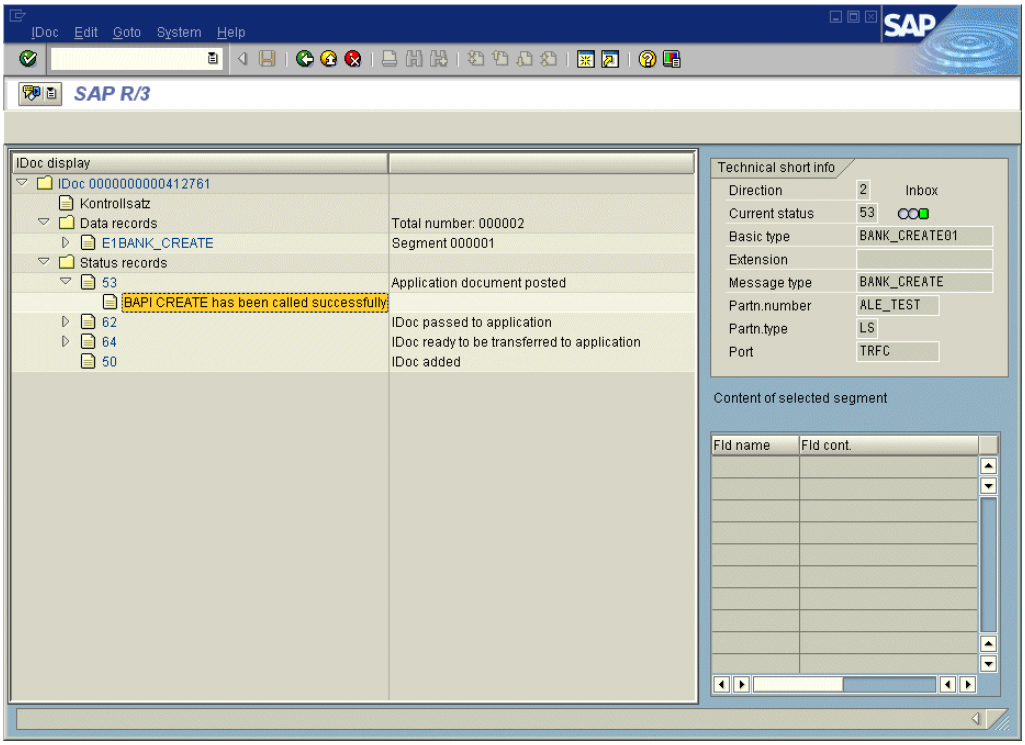
When executing the report, you have to enter a RECEIVER. This should be the name of the logical system you have defined for your SAP Adapter.

When executing the report in the SAP system, the SAP Adapter will transmit the XML document via HTTP (names of logical systems depend on your system configuration). The XML would look like this:

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
    <header>
      <delivery>
        <message>
          <messageID>0A125F1315B3A11B00000035</messageID>
          <sent>2000-06-07T09:22:40</sent>
        </message>
        <to>
          <address>urn:sap-com:logical-system:CERCLNT750</address>
        </to>
        <from>
          <address>urn:sap-com:logical-system:CERCLNT800</address>
        </from>
```

```
      </delivery>
    </header>
    <body xmlns="">
      <doc:Bank.Create xmlns:doc="urn:sap-com:document:sap:business">
        <BankKey>34981243</BankKey>
        <BankCtry>DE</BankCtry>
        <BankAddress>
          <BANK_NAME>Demo Bank</BANK_NAME>
          <REGION>BW</REGION>
          <STREET>Neurottstr. 16</STREET>
          <CITY>Walldorf</CITY>
          <SWIFT_CODE>ABCDDE12</SWIFT_CODE>
          <BANK_GROUP>SB</BANK_GROUP>
          <POBK_CURAC></POBK_CURAC>
          <BANK_NO>12345678</BANK_NO>
          <POST_BANK></POST_BANK>
          <BANK_BRANCH></BANK_BRANCH>
          <ADDR_NO>123</ADDR_NO>
        </BankAddress>
      </doc:Bank.Create>
    </body>
  </biztalk_1>
```

You can check the correct processing of your BAPI-call in the IDoc status monitor.



In the target SAP system, you can also inspect the correct processing using the ALE status monitor (BD87 in SAP release 4.6C. IDoc lists can also be displayed with WE05). If any application errors occurred, they are listed in the IDoc monitor.

webMethods

CHAPTER 12

Security

SAP Adapter Configuration

Please refer to the *webMethods Certificate Toolkit User's Guide* and the *webMethods Integration Server Administrator's Guide* for information on how to configure your Integration Server securely.

User Authentication Between the SAP Adapter and an SAP System

Authentication Through User Name and Password

When logging on through an HTTP or FTP client, standard user/password authentication is used. The user is mapped to an Integration Server session, and a service calling an SAP system is executed. This service uses the logon parameters associated with an RFC connection that likely does not reflect the identity of the original HTTP client. The user configured for the RFC Connection is usually a pool user shared among several physical users. This pooling allows for optimal performance.

If your RFC connection is configured to use SNC, a secure connection to the SAP system will be established. You need to install and configure the correct *sapcrypto.* library for your platform. This library supports SAP's Secure Network Communication (SNC) Standard. SNC works on top of the RFC protocol.

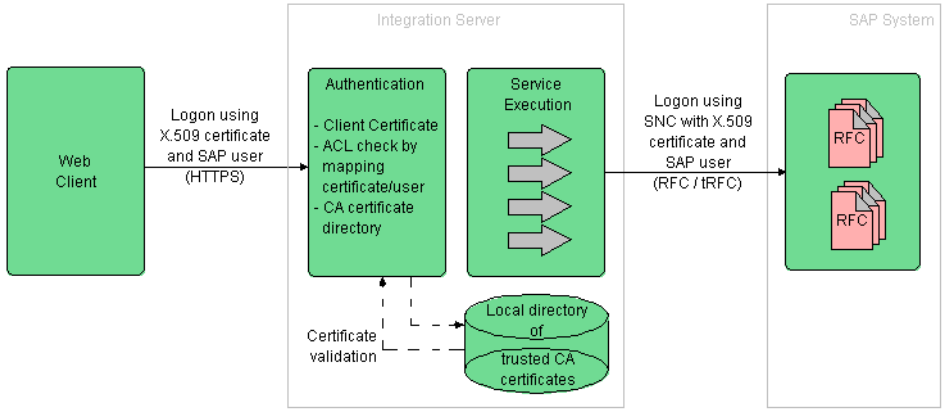
Authentication Through X.509 Certificate

Another method for user authentication in the Integration Server is through client authentication as a part of the SSL protocol. This requires that the corresponding HTTPS listener (port) requests a Client Certificate and that the client sends a trusted certificate that is mapped to an existing Integration Server user. A certificate is considered "trusted" if it has been issued by a CA (Certificate Authority) and is listed in a local CA Certificate Directory.

It is then possible to logon to the SAP system by means of this X.509 certificate. You need to install and configure a library supporting SAP's Secure Network Communication (SNC) Standard. SNC works on top of the RFC protocol. The following instructions describe the setup of this authentication method.

See "Using the SAP Adapter with the SAP Cryptographic Library for SNC" on page 203 for more information about SNC and SAP Adapters.

User Authentication via X.509 Certificate



Important! For the authentication via certificates against an SAP system, it is required to **enable SNC connections** for the RFC Connection defined at the SAP Adapter and on the SAP system. These settings are listed in the section "Configuring Adapter Connections" on page 56. For detailed information on the SAP system and RFC client settings for SNC see the corresponding SAP documentation.

Important! The HTTPS port defined on the Integration Server should have the **Request Client Certificate** option set for the **Client Authentication** field.

Tip! For more information on ports see the *webMethods Integration Server Administrator's Guide*.

If you want to log on to an SAP system via an Integration Server using any SAP user and a certificate, you can do so by providing a trusted certificate for the Integration Server.

Important! Before you can log on to the Integration Server using a trusted certificate, you have to import the (personalized) client certificate for each user from a local directory to the Integration Server and map it to an Integration Server user.

For validation purposes, you must also enter the path to the CA Certificate directory. The CA Certificate directory specifies the name of your local directory containing the root certificates of CAs that this server trusts. You may specify the directory using an absolute path or one that is relative to the *webMethods_directory*\IntegrationServer directory.

When a user logs on (for example, from a Web client) using this certificate, the Integration Server verifies the root certificate in the CA Certificate directory and then passes the client certificate, including the user name, to the SAP system. However, you must make sure that this user can access the services he wants to execute in the Integration Server. That is

why you must map the client certificate to the corresponding Integration Server user, or alternatively to a (standard) Integration Server user, depending on the authorizations required. If you want to execute a protected service within the Integration Server, the mapped user must be allowed in the corresponding ACLs (access control lists). For more information on ACLs, see the *webMethods Integration Server Administrator's Guide*.

Example:

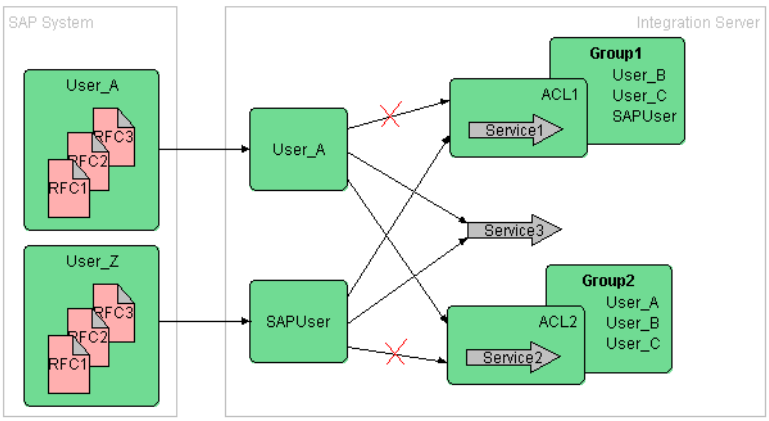
You want to execute a service on the Integration Server that retrieves sales order data from an SAP system. This service is protected by an ACL. The Integration Server user 'Sales' is registered in this ACL and allowed to execute the service. If you map the certificate to the user 'Sales', your SAP user can also execute the service. In addition, the SAP user must be authorized to execute the function modules of the corresponding function group.

To restrict the rights of the SAP logon users you should create specific user accounts in the SAP system with the minimum necessary set of authorizations. If for instance the SAP Adapter is used as a pure RFC-Server, it will only perform very few function callbacks to the calling SAP system. These callbacks are needed to determine the function interface specification. To allow for this it is sufficient to use an SAP logon user with the authorization to the following SAP standard function groups: RFC1, SDIF, SG00, SRFC.

If this user shall be used to call other application interfaces as well you need to add the respective function groups to the authorization list. Add this authorization to the standard authorization object 'S_RFC' and create an authorization profile which only contains this authorization. When creating the SAP user you can then assign this profile to it. For more details on authorization for SAP users please refer to the SAP documentation.

Authentication When the SAP Adapter Acts As an RFC-Server

Mapping SAP User to Integration Server User



If the SAP Adapter is called from an SAP system, the call is always trusted. Therefore, only the user name from the SAP system is used for the logon. This user is the user who triggered the synchronous or asynchronous call. If a user wants to execute a service protected by an ACL (access control list), this user must be entered in the corresponding ACL that allows access to this service.

Inbound calls can be secured by SNC the same way as outbound calls. To secure your inbound connection, enable SNC for your listener. See "Listeners" on page 87 for the SNC parameters you need to configure for a secure network connection.

Tip! In the user concept, the default user is called SAPUser (Password: 22101999). If the SAP Adapter is called by an SAP user that does not exist within Integration Server, the system switches automatically to the user SAPUser as the default user.

The User SAPUser is part of the User Group SAPUsers. It figures in the SAPUsers ACL that prevents unauthorized access to all Listener Notifications and Inbound Processes of the transports related to an SAP system.

The user called SAPUser, the User Group, and the ACL are created automatically when you start the Integration Server and SAP Adapter for the first time.

If an SAP user has been created in Integration Server in order to execute all Listener Notifications within Integration Server, you have to assign this user at least to the User Group SAPUsers.

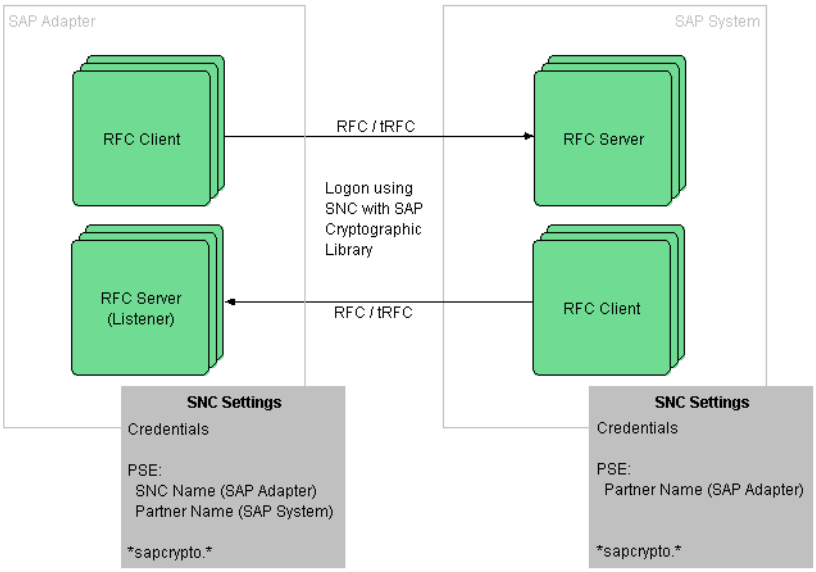
Important! If you want to avoid that an SAP user which has not been created within an Integration Server, can use the whole authorization range of the SAPUser, you should assign this SAP user individually to the corresponding User Group(s), respectively to the corresponding ACL(s).

Important! The user names in the Integration Server are case sensitive.

Using the SAP Adapter with the SAP Cryptographic Library for SNC

The SAP Cryptographic Library is the default security product delivered by SAP for performing encryption functions in SAP systems. You can use it for providing *Secure Network Communications* (SNC) between SAP system components. The `sapcrypto.lib` is also used to implement *Single Sign-On* (SSO). This section describes the procedure steps that are required to run the SAP Adapter communication using the `sapcrypto.lib`. For more detailed information on the installation of the `sapcrypto.lib` and the generation of PSE (*Personal Security Environment*) see *Using the SAP Cryptographic Library* for SNC in the SAP Online Documentation (SAP Library).

SAP Adapter Communication Using the SAP Cryptographic Library for SNC



To use an SAP Adapter with the SAP Cryptographic Library

1 Install the SAP Cryptographic Library (see the SAP Online Documentation)

2 Generate a PSE (Personal Security Environment) for SNC (see the SAP Online Documentation)

3 Create the Server's Credentials using SAPGENPSE (see the SAP Online Documentation)

4 Exchange the security information between the communication partners (servers): see the SAP Online Documentation.

5 Configure your SAP Adapter and the corresponding SAP system for SNC use (see Security Options in the section "Configuring Adapter Connections" on page 56 and "Listeners" on page 87).

6 Optional: Adopt the SECUDIR environment variable (see the SAP Online Documentation) to point to the correct location of your PSE files by modifying the startup script:

- Windows platforms: insert a line in server.bat as follows: after the SETLOCAL line set SECUDIR=<pse-path>
- Unix platforms: insert a line in server.sh as follows at the very beginning export SECUDIR=<pse-path>

Installing the SAP Adapter According to Your Security Policy

The SAP Adapter can only access SAP systems for which an RFC connection alias has been created. There is no service available that allows you to execute RFC calls to SAP systems that are not defined there.

In addition to this restriction, you can also protect access to SAP systems in an intranet by installing an additional firewall between the Integration Server and the SAP systems or putting the Integration Server in the DMZ. You can configure the firewall to restrict which SAP systems can be accessed from the SAP Adapter through the SAP router.

Finally you might even want to completely disallow an Integration Server in the DMZ to actively open connections to an SAP system in the intranet. To do so, you need to install two Integration Servers, one in the DMZ and one in the intranet. The Integration Server in the DMZ can than be configured as a reverse invoke server. Thus the Integration Server in the intranet has to establish the connection to the reverse invoke server whereas the data still flows synchronously from the outside to the inside. For information on how to configure reverse invoke please see the *webMethods Integration Server Administrator's Guide*.

webMethods

CHAPTER 13

Managing the DDIC Cache

Data DICtionary Cache (DDIC Cache)

The Data DICtionary Cache (DDIC) is a cache that holds information about SAP function modules, structure definitions, Business Objects, ALE mappings, and IDocs. The SAP Adapter retrieves this information from an SAP system when it performs RFC, BAPI, or IDoc lookups. To improve performance, the SAP Adapter caches information it receives about function modules and structure definitions.

The SAP Adapter receives information about function modules and structure functions when:

- An RFC, BAPI, or IDoc lookup is requested from the Lookup screen
- The SAP Adapter executes a service that invokes an RFC or BAPI on an SAP system
- An SAP system executes an RFC that invokes a service
- An IDoc gets processed by the SAP Adapter.

The SAP Adapter keeps separate cached function modules, structure definitions, Business Objects, ALE mappings, and IDocs for each SAP system. When the SAP Adapter requires specific data, it checks its DDIC cache for the specific SAP system to determine if the information it requires is in cache. If the required information is in the DDIC cache, the SAP Adapter uses the cached information rather than retrieving it from the SAP system.

The DDIC cache is always active. There are no configuration tasks required to activate it. When you are developing services, there may be times when the information in the DDIC cache becomes outdated; for example, if you change an RFC signature on an SAP system. In these situations, you can use the DDIC cache screens to view the information in the cache and remove specific function modules or structure definitions from the cache as necessary without having to restart the Integration Server.

Note: The DDIC cache does not persist through shutdown and restart of the Integration Server.

Viewing Information in the DDIC Cache

Perform one of the following procedures to view information about the function modules, structure definitions, Business Objects, ALE mappings, and IDocs that are in the DDIC cache.

To view information in the DDIC cache

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 Click DDIC-Cache.

The SAP Repository Cache table appears, showing all of the element types stored in the cache.

Note: The System ID field uniquely identifies an SAP system within one SAP domain. If one or more RFC connections use the same system ID, they will share the same cache. For more information about defining RFC connections, refer to "Configuring Adapter Connections" on page 56.

3 To view the names of cached function modules for an SAP system, click the number in the Functions column for the appropriate System ID.

The Cached Functions table appears. To view more details about a function module:

a Click the name of the function module for which you want to see the function interface.

The SAP Adapter displays a screen that lists the function interface for the selected function module.

b View the structure definition of the displayed parameters by following the hyperlink in the Table column.

4 To view the names of the cached structure definitions for an SAP system, click the number in the Structures column for the appropriate System ID.

The Cached Structures table appears. To see the structure definition, click the structure name of the structure you want to view.

The SAP Adapter displays a screen that lists the structure definition for the selected function module.

5 To view the names of the cached business objects for an SAP system, click the number in the Business Objects column for the appropriate System ID.

The Cached Business Objects table appears.

6 To view the names of the cached BAPIs for an SAP system (for which ALE information of an SAP system has been checked and extracted), click the number in the ALE Mappings column for the appropriate System ID.

The Cached ALE Mappings table appears.

7 To view the names of the cached IDocs for an SAP system, click the number in the IDocs column for the appropriate System ID.

The Cached IDocs table appears.

Note: The list also contains BAPIs for which no ALE interface has been generated in the SAP system, but for which the availability has already been checked. This is because the SAP Adapter tries only once to retrieve ALE information. If this retrieval fails, the SAP Adapter retains this fact.

Removing Information from the DDIC Cache

You can clear the entire cache for an SAP system, or you can clear individual elements from the cache. This section describes how to do both.

Clearing the DDIC Cache for an SAP System

To clear the entire DDIC cache for an SAP system

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 Click DDIC-Cache.

The SAP Repository Cache table appears, showing all of the element types stored in the cache.

3 Identify the System ID of the SAP system for which you want to clear the cache, and click the  icon in the Clear Cache column for that System ID.

Clearing Elements from the DDIC Cache

If a function module, structure definition, Business Object, ALE mapping, or IDoc in the DDIC cache becomes outdated or corrupt, you can use the following procedures to remove it from the DDIC cache.

To remove elements from the DDIC cache

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 Click DDIC-Cache.

The SAP Repository Cache table appears, showing all of the element types stored in the cache.

To clear an element from the DDIC cache:

a Identify the System ID of the SAP system for which you want to clear elements of the cache,.

b For that System ID, click the number in the column of the element type you want to clear. For example, if you want to clear a function module, click the number in the Functions column for the appropriate System ID.

The appropriate table for that element type appears.

c Identify the specific element you want to clear, and click the  icon in the Remove column for that element.

3 Click Return to DDIC-Cache to return to the SAP Repository Cache table.

webMethods

APPENDIX A

Package Contents

Package Layout

Apart from the native libraries, all of the files relevant to the SAP Adapter and its persistent state information are in the <*Integration Server*>/packages/WmSAP directory. The following table describes the directories and files that make up the SAP package directory.

Directory	Contents
code	Contains the SAP Adapter libraries, the JCo and SAP IDoc Class library jars, and sample source code.
config	Contains the following configuration file: cbr.cnf (content-based routing) — This file contains the configuration parameter for content-based routing.
logs	Contains the SAP Adapter log files.
ns	Contains namespace information for the SAP Adapter services.
pub, templates	Contains files and images comprising the SAP Adapter Administration user interface. Also contains all user and API documentation.

webMethods

APPENDIX B

Server Configuration

General SAP Adapter Settings

For the general settings of the SAP Adapter, you can use the Administrator User Interface.

To change the general settings

1 In the Adapters menu in the Integration Server Administrator navigation area, click SAP Adapter.

2 Click Settings.

3 You can edit the following:

Setting	Description
Timeout (minutes)	Delay (minutes) until an unused repository connection to an SAP system is timed out (default: 5)
Timeout check period (seconds)	Time interval in seconds between checks whether unused pooled repository connections have timed out
Sessionpool size	Maximal number of repository connections in one RFC connection pool to one SAP system (default: 10)
Poolqueue waiting time (seconds)	Delay (seconds) until requests waiting for a repository connection from a pool time out in the queue
Check time (minutes)	Time interval (minutes) between checks of the listener. It is checked whether the RFC Handle is still valid. If the Gateway is running, any inactive Listener is restarted at the latest after this interval.
Response time (seconds)	Delay (seconds) until a listener responds at the latest to an incoming request.
Default XRFC version	Version of RFC-XML sent. Valid versions: 0.9 and 1.0 (default).
SNC library path	Path of the SNC library needed for secure RFC connections.

Setting	Description
Log Throughput data	Flag to decide whether to write basic throughput information to log. Valid values: “On” and “Off” (default).
Use Monitor	Flag to decide whether performance monitoring should be used. Valid values: “On” and “Off” (default).

server.cnf

This file is created when you start the Integration Server for the first time. Some of the following statements can be added manually after installing the SAP Adapter.

Use the Edit Extended Settings feature of the Integration Server Administrator to view and edit the key statements. See the *webMethods Integration Server Administrator's Guide* for more information.

Tip! You can also add and edit key statements by shutting down the Integration Server and manually editing the *webMethods_directory*/IntegrationServer/config/server.cnf file.

The following statements are added automatically when Integration Server and SAP Adapter are started for the first time:

Setting	Description
watt.sap.repo.maxPooledConnections	Maximal number of repository connections in one RFC connection pool to one SAP system (default: 10)
watt.sap.repo.timeout	Delay (minutes) until an unused repository connection to an SAP system is timed out (default: 5)
watt.sap.repo.timeoutCheckPeriod	Time interval in seconds between checks (seconds) whether unused pooled repository connections have timed out
watt.sap.repo.maxWaitForPool	Delay (seconds) until requests waiting for a repository connection from a pool time out in the queue
watt.sap.listener.checkTime	Time interval (minutes) between checks of the listener. It is checked whether the RFC Handle is still valid. If the Gateway is running, any inactive Listener is restarted at the latest after this interval.

Setting	Description
watt.sap.listener.responseTime	Delay (seconds) until a listener responds at the latest to an incoming request.
watt.sap.debug.level	Debug level from 1 to 10
watt.sap.debug.facList	List of Logging Facilities for the WmSAP Package.

Change/add the following statements to the server.cnf to increase performance:

Setting	Description
watt.sap.rfcxml.version	Version of RFC-XML (XRFC) sent. Valid versions: 0.9 and 1.0 (default).
watt.sap.idocxml.escaping	Version of illegal character escaping in IDoc-XML. Valid versions: 4.6 and 5.0 (default)
watt.sap.xml.prettyPrint	Flag that indicates if the SAP XML dialects should be rendered pretty printed. The default is “true”. Set the value to "false" to remove all unnecessary whitespace and thus improve performance.
watt.sap.xtn.fastAsyncMode -	Setting it to "true", enables the synchronous write cache. Setting it to "false" disables the cache. This property works together with the watt.sap.xtn.cacheFlushPeriod and watt.sap.xtn.cacheTimeToLive - properties. Setting this feature to “On” will force the transaction manager not to persist each update of an transaction but rather keep the state changes in memory and instead persist it periodically.
watt.sap.xtn.cacheFlushPeriod	It specifies the time period in minutes, how often the TransactionManager will flush yet unsaved changes in the transaction Cache to the file system. Default: every 5 minutes.
watt.sap.xtn.cacheTimeToLive -	The transaction cache works both as read and write cache. After a transaction is read the first time, it will be kept in the cache for a specific amount of time (cacheTimeToLive). If the transaction was not accessed during this time, it will be removed from the cache after this period has ended (-) in order to keep the cache from growing indefinitely.

Change/add the following statements to the server.cnf to improve security:

Setting	Description
watt.sap.snclibpath	Path of the SNC library needed for secure RFC connections.

Change/add the following statements to the server.cnf to specify monitoring options:

Setting	Description
watt.sap.throughput	Flag to decide whether to write basic throughput information to log. Valid values: true and false (default)
watt.sap.monitor	Switch to turn Component Responsetime Monitoring 'On' (default) or 'Off'

Change/add the following statements to the server.cnf to specify miscellaneous options:

Setting	Description
watt.sap.aclset	If this property in not available or not set to “true”, then during the next startup of the Integration Server, all SAP Adapter-relevant ACL settings will be reset to their default values.
watt.sap.systat01.partnerNumber	To send SYSTAT01 IDocs from the SAP Adapter back to the SAP system, the Integration Server must be registered as a logical system. By default, the name "BUSCON" is used. If you want to use a different name, you can override the default value using this property.

webMethods

APPENDIX C

ABAP Types in the SAP Adapter

ABAP Types

ABAP Type	Description	Dictionary Types	Java Type
C	character[s]	CHAR, CLNT, CUKY, LANG, LCHR, UNIT	java.lang.String
D	Gregorian Calendar date (yyyy-MM-dd)	DATS	java.lang.String
F	IEEE double-precision 64- bit floating point	FLOAT	java.lang.Double
I	4-byte signed integer	INT4	java.lang.Integer
N	numeric character[s]	NUMC, ACCP	java.lang.String
P	decimal number	CURR, DEC, QUAN	java.lang.String
T	time (HH.mm.ss)	TIMS	java.lang.String
X	binary octets	RAW	byte[]
b	1-byte unsigned integer	INT1	java.lang.Integer
s	2-byte signed integer	INT2	java.lang.Integer
g	variable length string	STRING	java.lang.String
y	variable length binary octets	RAWSTRING	byte[]

webMethods

APPENDIX D

Built-in Services

SAP Client Services

pub.sap.client:connect

Establishes a connection to an SAP system.

Input Parameters

serverName	Alias of the SAP system to which the connection should be established. The name must match a configured RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the default client is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

result	"Ok" if connection was established successfully.

Example

Use the pub.sap.client:connect service when you want to establish a connection to a target SAP system. This might be useful for creating a session pool. In the most common scenarios, it is not necessary to invoke this service explicitly, as it is invoked implicitly (for example, when a pub.sap.client:invoke or pub.sap.client:invokeTransaction is issued).

pub.sap.client:lockSession

Locks an RFC connection to your Integration Server session so that you will always exclusively use this RFC connection for subsequent calls.

Input Parameters

serverName	Alias of the SAP system on which the session will be locked. The name must match a configured RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the user specified in the SAP system alias settings is used.
$pass	Optional. Password for the session. If no password is specified, the password specified in the SAP system alias settings is used.
$language	Optional. Language used during the session. If no language is specified, the language specified in the SAP system alias settings is used.

Return Values

None.

Example

This service locks a session to trigger a `commit work` command inside the SAP system. This causes a database commit and the start of the processing of posted data.

Note: This service only works with SAP systems version 4.0A and higher because BAPIs do not write data directly to the database but use the posting engine inside the SAP system.

The data will not be written to the database until the client triggers a `commit work` command. In order to call a BAPI that writes data, you must perform the following steps on the Integration Server:

1 Call the pub.sap.client:lockSession service in order to get an exclusive connection to the SAP system.

2 Perform the BAPI calls.

3 Call the pub.sap.bapi:commit or pub.sap.bapi:rollback service

4 Call the pub.sap.client:releaseSession service in order to release the exclusive connection to the SAP system and clean up used resources on the Integration Server.

pub.sap.client:releaseSession

Releases a locked session on an SAP system.

Input Parameters

serverName	SAP system alias on which the locked session is located. The name must match a configured RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

There are no return values

Example1

The service will release a session after a `commit work` or `rollback work` command has been triggered inside the SAP system.

Note: This service only works with SAP systems version 4.0A and higher because BAPIs do not write data directly to the database but use the posting engine inside the SAP system.

In order to call a BAPI that writes data, you must perform the following steps on the Integration Server:

1 Call the pub.sap.client:lockSession service in order to get a exclusive connection to the SAP system.

2 Perform the BAPI calls.

3 Call the pub.sap.bapi:commit or pub.sap.bapi:rollback service

4 Call the pub.sap.client:releaseSession service in order to release the exclusive connection to the SAP system and clean up used resources on the Integration Server.

Example2

You also need this service if you want to lock and change a certain database object. To do this, please proceed as follows:

1 Call the pub.client:lockSession service in order to get an exclusive connection to an SAP system.

2 Lock an object by calling a BAPI_ *_ENQUEUE.

3 Make your changes by invoking other BAPIs.

4 Release the object by calling a BAPI_*_DEQUEUE.

5 Call the pub.client:releaseSession service.

pub.sap.client:invoke

Invokes an RFC function module in synchronous mode on a given SAP system.

Input Parameters

This service also needs the inputs (imports and tables) required by the function module.

serverName	SAP system alias on which the function module will be invoked. The name must match a configured RFC connection alias at the SAP Adapter.
$rfcname	Name of the function module to be invoked.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

This service returns the outputs (exports and tables) returned by the function module.

$rfctime	Time (ms) spent within the RFC library to complete the invocation.
$runtime	Total invocation time (ms), including processing time within the SAP Adapter.

$encoding	MIME-compliant character set corresponding to the session's SAP code page.
$call	Flag indicating whether the pipeline represents a request (true) or a response (false) of a function module

Example

This service is used when you are directly calling an RFC on an SAP system without creating an RFC adapter service. (When an RFC adapter service is used, this service is invoked implicitly.)

pub.sap.client:invokeTransaction

Invokes an tRFC function module on a given SAP system.

Input Parameters

serverName	SAP system alias on which you want to invoke the function module. The alias must match a configured RFC connection alias at the SAP Adapter.
$rfcName	Name of the function module to be invoked.
$tid	Transaction this invoke is assigned to.
$queueName	Optional. Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

$runtime	Total invocation time (ms), including processing time within the SAP Adapter.
$rfctime	Time (ms) spent within the RFC library to complete the invocation.
$encoding	MIME-compliant character set corresponding to the session's SAP code page.
$call	Flag indicating whether pipeline represents a request (true) or a response (false) of a function module.

Example

This service is used in outbound flows and can handle both synchronous and transactional RFC calls. Therefore, it can be seen as a generic wrapper for pub.sap.client:invoke, pub.sap.client:invokeTransaction and pub.sap.client:confirmTID.

pub.sap.client:createTID

Call this service if you want to obtain a transaction ID (TID; which is a GUID) that conforms to the format of SAP TIDs. The obtained TID can be used as an input value for pub.sap.client:invokeTransaction, pub.sap.client:sendIDoc, and pub.sap.client:confirmTID.

Important! If no TID exists in the pipeline, then the pub.sap.transport.ALE:OutboundProcess service performs the function of both the pub.sap.client:createTID and pub.sap.client:invokeTransaction services. For IDocs, it is recommended that you use either the pub.sap.transport.ALE:OutboundProcess service or the call sequence of pub.sap.client:createTID and pub.sap.client:sendIDoc.

Input Parameters

serverName	Alias of the SAP system that sends the TID. This name must match a configured RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

$tid	Contains the TID.

Example

This service is used in tRFC client scenarios, which require a transactional ID (TID) uniquely identifying the transaction. See also pub.sap.client:confirmTID.

pub.sap.client:confirmTID

Confirms the transaction specified by the TID on the specified SAP system. This service must be called after a transactional invoke in order to complete a transaction on the target system.

Input Parameters

serverName	SAP system alias on which the transaction should be confirmed. The name must match a configured RFC connection alias at the SAP Adapter.
$tid	Transaction ID to be confirmed.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

None.

Example

This service is used in tRFC client scenarios in order to confirm a transaction which has been executed completely.

1 Invoke pub.sap.client:createTID. This service creates an SAP-conformant TID. It is necessary to have one, otherwise you cannot invoke the function module with

pub.sap.client:invokeTransaction. If you have already obtained a TID, you should use that one in order to guarantee that the transaction is executed only once.

2 Invoke pub.sap.client:invokeTransaction. This service invokes the given function module as a tRFC on the SAP system specified by *serverName*. The TID is passed as a parameter on this call.

3 Invoke pub.sap.client:confirmTID. Confirms that the transaction has completed. Pass the same *serverName* and *$tid* value on this call.

pub.sap.client:getAttributes

Connects to the given SAP system and returns information specific to one RFC client connection session:

Input Parameters

serverName	Alias of the SAP system from which you want to receive the RFC connection-specific attributes. The name must match a configured RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

Document named *rfcAttributes* containing specific information about one RFC connection session:

destination	String. Destination of the connection.
ownHost	Hostname (or IP) of the machine the Integration Server is running on.
partnerHost	Hostname (or IP) of the machine the SAP system is running on.
systemNumber	System number of the SAP system.
systemID	Unique three-letter-ID of SAP system (in local network).
client	Client the session is connected to.

user	SAP user that has connected with this session.
language	Logon language.
ISOLanguage	ISO name for logon language.
ownCodepage	SAP codepage this connection is using.
partnerCodepage	SAP codepage the SAP system is running on.
ownRelease	Version of the loaded RFC library used by the SAP Adapter.
partnerRelease	Release of the SAP system.
kernelRelease	Kernel release of the SAP system.
partnerType	RFC type of the partner; R/3 (3), R/2(2) or external RFC server (E).
trace	Flag indicating whether trace is turned on(true) or off(false) for this connection.
ownType	RFC type of the SAP Adapter; should always be E.
rfcRole	Role of the SAP Adapter in this call; should always be: C (for client).
CPICConversationID	CPIC ID of the connection (low level protocol information).
encoding	IANA-encoding that is equivalent to the SAP codepage used, for example: ISO-8859-1.
charset	Charset that is equivalent to the SAP codepage used, for example: ISO8859_1 (used in Java constructors).
bytesPerChar	Number of maximum bytes used per char in the codepage.

pub.sap.client:getFunctionInterface

Looks up the function interface definition of an RFC function module in the given SAP system.

Input Parameters

$rfcname	Function module for which the interface definition is looked up.
serverName	Alias of the SAP system on which the RFC function module is defined. The name must match a configured RFC connection alias at the SAP Adapter.

Return Values

Document list named *params* that contains all parameter-related information:

paramClass	Class of parameter; I (importing), E(exporting), T(table), X(exception).
parameter	Name of the parameter in the function interface.
tabName	Name of the table in the SAP system the parameter is referring to.
type	Type of the parameter (for example: CHAR, STRUCTURE, TABLE, ...)
length	Internal length in bytes of the parameter.
decimals	Only relevant for decimal types; number of decimals used.
optional	Flag indicating whether parameter is optional.

pub.sap.client:getStructureDefinition

Looks up the structure definition of an SAP structure in the given SAP system.

Input Parameters

structName	Structure name for which the definition should be looked up.
serverName	Alias of the SAP system on which the RFC structure is defined. The name must match a configured RFC connection alias at the SAP Adapter.

Return Values

Document list named *fields* that contains all field-related information.

tabLength Length of the structure when used in a table. This might differ from the sum of all field lengths, as the number types have to be aligned to memory segments.

fields

Key	Description
fieldName	Name of the field.
tabName	Optional. Name of the table/structure in the SAP system the field is referring to.
position	Position within structure.
offset	Offset in bytes to this field.
length	Internal length in bytes.
decimals	Only relevant for decimal types; number of decimals used.
type	Type of the field (for example: CHAR, STRUCTURE, TABLE, ...).

pub.sap.client:sendIDoc

Sends an IDoc to a given SAP system.

Input Parameters

serverName	Alias for the SAP system that will receive the IDoc. The alias must match a configured RFC connection alias at the SAP Adapter.
iDocList	Contains the IDoc(s) as object of the com.sap.mw.idoc.IDoc.DocumentList type.
$tid	Transactions this invoke is assigned to.
$queueName	Optional. Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

$runtime	Total invocation time (ms), including processing time within the SAP Adapter.
$rfctime	Time (ms) spent within the RFC library to complete the invocation.
$encoding	MIME-compliant character set corresponding to the session's SAP code page.
$call	Flag indicating whether pipeline represents a request (true) or a response (false) of a function module.

pub.sap.client:ping

Checks the response time of an SAP system.

Input Parameters

serverName	RFC connection alias for the SAP system you want to ping. The name must match a configured RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values:

responseTime	Response time of the SAP system.

XRFC Services

pub.sap.rfc:decode

Converts an RFC-XML (XRFC) string into the pipeline so that is in a format that can be passed to an SAP system.

Input Parameters

xmlData	String. An RFC-XML message.
bytes	Optional. Byte array that contains the data to be decoded as XRFC.
node	Optional. XML object that represents the XRFC data (you will get a node when putting an XML file via FTP with extension .xml). The service checks in that order if an input document is available: xmlData, bytes, node.

Return Values

$encoding	Optional. Specifies the encoding used in the input document's XML header, (for example: iso-8859-1) if $encoding has not been included in the pipeline before.

Example

This service can be used when an RFC-XML document is posted to the Integration Server in a variable.

pub.sap.rfc:encode

Converts an RFC call stored in the pipeline to an XML string in a format specified by the SAP RFC-XML (XRFC) specification.

Input Parameters

This service needs the RFC calls in Values representation as input.

serverName	SAP system used as a repository. This name must match a configured RFC connection alias at the SAP Adapter.
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1

$call	Optional. This flag specifies whether the pipeline should be interpreted as a call (true) or as a response (false)
$envelope	Optional. XRFC, bXML, or SOAP. Specifies the type of document generated.

Return Values

xmlData	String. Contains the RFC-XML representation of the RFC.
contentType	The content type associated with the generated document (e.g. application/x-sap.rfc)

Example

Use this service to create RFC-XML corresponding to the pipeline and perhaps forward it to another server (for example, via HTTP). This is done in the XML transport's Outbound Process for function modules.

pub.sap.rfc:createTemplate

Creates an RFC-XML (XRFC) template for the specified function module

Input Parameters

serverName	SAP system alias that is used as a repository. The alias must match a configured RFC connection alias at the SAP Adapter.
$rfcname	Optional. Function module for which you want to create the template.
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1
$call	This flag specifies whether a call template (true) or a response template (false) is created.
$envelope	Optional. XRFC, bXML, or SOAP. Specifies the type of template generated.

Return Values

xmlData	String. Contains the XML template that represents the RFC.

Example

Use this service to create templates for testing purposes.

IDoc Services

Note: Due to the integration of the SAP Java IDoc Class Library into SAP Adapter 6.5, some service parameters in this part have changed. They may contain the additional parameter iDocList (a com.sap.mw.idoc.IDoc.DocumentList object) that replaces or extends the parameters IDOC_CONTROL(_REC_40) and IDOC_DATA(_REC_40).

pub.sap.idoc:encodeSDATA

Converts every SDATA field from a Document object to a String object. This service is usually called prior to sending an IDoc to an SAP system via pub.sap.client:invokeTransaction or by an RFC adapter service for function module "IDOC_INBOUND_ASYNCHRONOUS" or "INBOUND_IDOC_PROCESS".

Note: There is also a client service called pub.sap.client:sendIDoc that allows you to send the iDocList directly.

Input Parameters

serverName	SAP system alias which is used as repository for structure information about the IDoc. The name must match a configured RFC connection alias at the SAP Adapter.
IDOC_CONTROL *IDOC_DATA*	Both keys are Document lists (Tables) containing the control and data tables for the IDoc.
- OR -	
IDOC_CONTROL_REC40 *IDOC_DATA_REC40*	The SDATA field is a Document object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field). Note: This service handles both IDoc versions 2 and 3. The difference between the two is that, for IDocs version 2, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 3, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.
iDocList	Contains the IDoc(s) as object of com.sap.mw.idoc.IDoc.DocumentList.
- OR -	
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1

Return Values

IDOC_CONTROL *IDOC_DATA*	Both keys are document lists (Tables) containing the control and data tables for the IDoc.
- OR -	The SDATA field is a 1000-byte field.
IDOC_CONTROL_REC40 *IDOC_DATA_REC40*	At this point, the IDoc in the pipeline is ready to be sent to an SAP system via either pub.sap.client:invokeTransaction or an adapter service for one of the function modules (IDOC_INBOUND_ASYNCHRONOUS" or "INBOUND_IDOC_PROCESS").

pub.sap.idoc:decodeSDATA

Converts the SDATA field from a String object to a Document object and immediately converts the whole list of IDocs to a com.sap.mw.idoc.IDoc.DocumentList object for further processing. With Release 6.5, the service is only required if you receive an IDoc in the tables representation from a sending system. That is, you have created an RFC adapter notification for function module "IDOC_INBOUND_ASYNCHRONOUS" or "INBOUND_IDOC_PROCESS".

Input Parameters

serverName	SAP system alias which is used as repository for structure information about the IDoc. The name must match a configured RFC connection alias at the SAP Adapter.
IDOC_CONTROL *IDOC_DATA*	Both keys are Document lists (tables) containing the control and data tables for the IDoc.
- OR -	At this point, the SDATA field of each segment consists of a 1000-byte field.
IDOC_CONTROL_REC40 *IDOC_DATA_REC40*	Note: This service handles both IDoc versions 2 and 3. The difference between the two is that, for IDocs version 2, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 3, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.

Return Values

iDocList	Contains the IDoc(s) as object of com.sap.mw.idoc.IDoc.DocumentList.

pub.sap.idoc:encodeString

Encodes an IDoc to a String equivalent, the format that is used by the file port of an SAP system. The service first checks for the iDocList field, if that parameter is not present, it will look for the IDOC_* record lists.

Note: When saving to a file, make sure that String is saved with correct encoding.

Input Parameters

serverName	SAP system alias which is used as repository for structure information about the IDoc. The name must match a configured RFC connection alias at the SAP Adapter.
iDocRelease	Optional. The SAP system release that the IDoc will be sent to.
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.
iDocList	Contains the IDoc(s) as object of com.sap.mw.idoc.IDoc.DocumentList.
- OR -	
IDOC_CONTROL *IDOC_DATA* - OR - *IDOC_CONTROL_REC40* *IDOC_DATA_REC40*	Both keys are document lists (tables) containing the control and data tables for the IDoc. The SDATA field is an orderly Document object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field). Note: This service handles both IDocs versions 2 and 3. The difference between the two is that, for IDocs version 2, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 3, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

Return Values

iDocString	String. The IDoc in the file port.

pub.sap.idoc:decodeString

Decodes an IDoc from a String equivalent, the format that is used by the file port of an SAP system.

Input Parameters

serverName	SAP system alias which is used as repository for structure information about the IDoc. The name must match a configured RFC connection alias at the SAP Adapter.
iDocString	String. The iDoc in the file port.
$encoding	Optional. Specifies the encoding used to decode the IDoc string, e.g. iso-8859-1.

Return Values

iDocList	Contains the IDoc(s) as object of com.sap.mw.idoc.IDoc.DocumentList.

pub.sap.idoc:iDocToDocument

Generates a Document representation of an IDoc list to perform mappings in Flow language.

Input Parameters

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
conformsTo	Name of the document structure of the IDoc. The document structure is used as a template to discriminate between documents and document lists. Without conformsTo, all tables in the hierarchical structure will be document lists.

Return Values

document	IDoc in hierarchical structure.

pub.sap.idoc:documentToIDoc

Complementary service to pub.sap.idoc:iDocToDocument.

Input Parameters

document	IDoc in hierarchical structure.

Return Values

iDocList	Contains the IDoc(s) as objects of the type com.sap.mw.idoc.IDoc.DocumentList.

pub.sap.idoc:iDocToTables

Generates the tables compatible with SAP Adapter version 4.6 from an object compatible with SAP Adapter version 6.5. Do this before sending the IDoc to an SAP Adapter version 4.6 inbound transport.

Input Parameters

iDocList	Contains the IDoc(s) as object of the type com.sap.mw.idoc.IDoc.DocumentList.

Return Values

IDOC_CONTROL *IDOC_DATA* - OR - *IDOC_CONTROL_REC40* *IDOC_DATA_REC40*	Both keys are Document lists (tables) containing the control and data tables for the IDoc. The SDATA field is an orderly Document containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field). Note: This service handles IDoc versions 2 and 3. The difference between the two is that, for IDoc version 2, the service creates IDOC_CONTROL and IDOC_DATA in the pipeline. For IDoc version 3, it creates IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

pub.sap.idoc:tablesToIDoc

Complementary service to pub.sap.idoc:iDocToTables.

Input Parameters

IDOC_CONTROL *IDOC_DATA* - OR - *IDOC_CONTROL_REC40* *IDOC_DATA_REC40*	Both keys are Document lists (tables) containing the control and data tables for the IDoc. The SDATA field is an orderly Document containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field). **Note:** This service handles IDoc versions 2 and 3. The difference between the two is that, for IDoc version 2, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDoc version 3, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

Return Values

iDocList	Contains the IDoc(s) as object of the type com.sap.mw.idoc.IDoc.DocumentList.

IDoc-XML Services

pub.sap.idoc:encode

Service that converts an IDoc list to an XML string in a format specified by the SAP IDoc-XML Specification.

Input Parameters

iDocList - OR -	Contains the IDoc(s) as object of the type com.sap.mw.idoc.IDoc.DocumentList.
IDOC_CONTROL *IDOC_DATA* - OR - *IDOC_CONTROL_REC40* *IDOC_DATA_REC40*	Both keys are Document lists (tables) containing the control and data tables for the IDoc. The SDATA field is an orderly Document containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field). Note: This service handles IDoc versions 2 and 3. The difference between the two is that, for IDoc version 2, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDoc version 3, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1

Return Values

xmlData	String. The IDoc in XML. The XML format is consistent with the SAP-XML specification.

Example

This service is most often used when you want to receive an IDoc from an SAP system and convert it to XML.

- Invoke pub.sap.idoc:encode. This takes the pipeline containing the IDoc as input and converts it to an XML string. The XML string is accessible in the pipeline via the xmlData key.

pub.sap.idoc:decode

Service that converts an XML string in a format specified by the SAP IDoc-XML Specification into an IDoc that is of the type com.sap.mw.idoc.IDoc.DocumentList and necessary for an RFC call (using pub.sap.client:sendIDoc).

Input Parameters

bytes	Byte array. Contains the data to be decoded as IDocXML.
- OR -	
node	XML Node object that represents the IDocXML data (you will get a node, e.g. when putting an XML file via FTP with extension .xml). The service checks in that order if an input document is available: xmlData, bytes, node
- OR -	
xmlData	String. The IDoc in XML. The XML format is consistent with the SAP IDoc-XML specification.

Return Values

$encoding	Specifies the encoding from the input document's XML header, e.g. iso-8859-1.
iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.

Example

This service is the first service called when you want to send an IDoc-XML document to an SAP system. The following is a sequence of service calls that take an IDoc in XML format, convert it to a com.sap.mw.idoc.IDoc.DocumentList object, then fire it into the SAP system:

1 Invoke this service (pub.sap.idoc:decode). This takes the input *xmlData* (the XML String) and creates an com.sap.mw.idoc.IDoc.DocumentList object that is in *RFC-ready form.* (This means that it matches the RFC call used to send an IDoc into the SAP system, and that the pipeline is almost in the required format).

2 Invoke pub.sap.client:createTID. Use this service to request a transaction id from an SAP system that can be used for sending the IDoc to the SAP system.

Note: Store this transaction ID, so that you can resend the IDoc in case of errors with the same ID.

3 Invoke pub.sap.client:sendIDoc.

Monitoring Services

pub.sap.monitor.aleaud01:swapSenderReceiverLS

Switches Sender and Receiver (logical systems) for a list of given ALEAUD01 IDocs.

Input Parameters

iDocList	List of ALEAUD01 IDocs to be switched.

Return Values

iDocList	List of ALEAUD01 IDocs with switched RVCPRN and SNDPRN fields.

pub.sap.monitor.idoc:trace

Internal service used to trace the status of certain IDocs back to either the sending SAP system or to a further processing SAP Adapter. This service is directly called from SAP systems.

Input Parameters

SELECTION_DATE	Date parameter passed to IDOC_DATE_TIME_GET on the sending SAP system.
SOURCE_SYSTEM	The partner number of the sending SAP system.
T_IDOCINFO	List of IDocs to be traced.

Return Values

T_IDOCINFO	List of IDocs with updated trace information.

pub.sap.monitor.systat01:report

This service sends out SYSTAT01 IDocs to the sending SAP system for all IDocs that have been received in 'Monitoring' mode. Can be executed exactly once for the IDocs of a transaction.

Input Parameters

None.

Return Values

None.

bXML Services

pub.sap.bapi:decode

Decodes bXML documents and generates a corresponding pipeline.

Input Parameters

xmlData	String. A document in bXML format.
- OR-	
bytes	Byte array. Contains the data to be decoded as bXML.
- OR -	
node	XML Node object that represents the bXML data (for example, you get a node when putting an XML file via FTP with extension .xml). The service checks in that order if an input document is available: xmlData, bytes, node.

Return Values

$encoding	Optional. Specifies the encoding found in the bXML document, e.g. iso-8859-1.

pub.sap.bapi:encode

Converts a BAPI call represented in a pipeline to an XML string in a format specified by the SAP bXML specification.

Input Parameters

objectName	Name of the business object.
bapiName	Name of the BAPI.
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.
$call	Optional. This flag specifies whether the pipeline should be interpreted as a call (true) or as a response (false). The default is “false”.

$abapexception	Optional. If this object is in the pipeline, an Exception document will be created.
$metabapi	Optional. Contains metadata information about the BAPI, that should be encoded.

Return Values

xmlData	A document in bXML format as a string.

pub.sap.bapi:createTemplate

Generates a bXML document template for the definition of a BAPI in an SAP system.

Input Parameters

serverName	SAP system alias for the SAP system that is used as a repository. The alias must match a configured RFC connection alias at the SAP Adapter.
objectName	Name of the business object to encode.
bapiName	Name of the BAPI.
$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.
$call	Optional. This flag specifies whether a call template (true) or a response template (false) is created.

Return Values

xmlData	The bXML template for the BAPI as a string.

BAPI Services

pub.sap.bapi:commit

Commits a transaction with a single or multiple BAPI calls.

Note: You need to use pub.sap.client:lockSession/releaseSession to make sure to keep your connection in transaction scenarios as the transaction context is stored with the session in the SAP system

Input Parameters

serverName	Alias of the SAP system on which the transaction should be committed. This name must match an RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the default client is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.
wait	Optional. Flag indicating if call should wait until data is really written to database.

Return Values

Return	Return code in BAPI style keeping the success/failure of the commit.

pub.sap.bapi:rollback

Rolls back a transaction with a single or multiple BAPI calls.

Note: You need to use pub.sap.client:lockSession or pub.sap.client:releaseSession to make sure to keep your connection in transaction scenarios as the transaction context is stored with the session in the SAP system.

Input Parameters

serverName	Alias of the SAP system on which the transaction should be rolled back. This name must match an RFC connection alias at the SAP Adapter.
$client	Optional. Client for the session. If no client is specified, the default client is used.
$user	Optional. User name for the session. If no user is specified, the default user is used.
$pass	Optional. Password for the session. If no password is specified, the default password is used.
$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

Return	Return code in BAPI style keeping the success/failure of the commit.

Transaction Administration Services

pub.sap.transaction:list

Returns a list of transactions from the transaction store.

Input Parameters

endDate	Optional. Filter criteria: Last creation date of the transaction. The format of this field should be the same as the one defined under Settings ▸ Logging ▸ Log Timestamp Format.
startDate	Optional. Filter criteria: Earliest creation date of the transaction. The format of this field should be the same as the one defined under Settings ▸ Logging ▸ Log Timestamp Format.
sender	Optional. Filter criteria: name of the sender of the transaction.
receiver	Optional. Filter criteria: name of the receiver of the transaction.
msgType	Optional. Filter criteria: name of the message type of the transaction.
state	Optional. Filter criteria: state of the transaction.
$sortKey	Optional. Specifies the sort key – valid: “noSort” [default], “$tid”, “sender”, “receiver” “msgType”, “date” and “state”. The resulting transaction list will be sorted by the given attribute.
$dir	Optional. Specifies the sort direction. “descending” triggers a descending sort order; any other value triggers ascending sort order.

Return Values

transactions	Optional. A Document List containing one document with detail information for each transaction. The following keys are always included: "$tid", "sender", "receiver", "msgType", "state", "date" (the date/time, when the state of this transactions was last changed, formatted in the usual "Log Timestamp Format").
$sortKey	Optional. Specifies the sort key – valid: “noSort” [default], “$tid”, “sender”, “receiver” “msgType”, “date” and “state”. The resulting transaction list will be sorted by the given attribute.
$dir	Optional. Specifies the sort direction. “descending” triggers a descending sort order; any other value triggers ascending sort order.

pub.sap.transaction:get

Returns detail information for a single transaction.

Input Parameters

$tid	The TID of the transaction

Return values

sender	Optional. Name of the sender of this transaction.
receiver	Optional. Name of the receiver of this transaction.
msgType	Optional. Name of the message type of this transaction.
state	Optional. Current state of this transaction.
date	Optional. Creation date of this transaction.
auditLog	Optional. A Document List containing the date and message that tracks of all events that occurred during processing of this transaction.

pub.sap.transaction:getMessageBody

Returns the pipeline containing the actual message body.

Note: This Service will only work if the storing of message bodies has not been disabled.

Input Parameters

$tid	The TID of the transaction

Return values

None.

pub.sap.transaction:log

Adds a new audit log entry to the given transaction, and optionally changes the transaction status to the given new state.

Input Parameters

$tid	TID of the transaction.
msg	Optional. Message string to be logged.
state	Optional. String. Valid values are: ■ Created ■ Executed ■ Committed ■ Rolledback ■ Confirmed

Return Values

None.

pub.sap.transaction:setCacheParameter

This services sets new runtime parameters for the cache of the transaction store.

Input Parameters

writeCache	Optional. If “On”, the write cache of the transaction will be switched on. Any other value will switch it off.
cacheTimeToLive	Optional. This value sets the maximum time-to-live for a cache entry (value in seconds). Valid range: 0-100.
cachFlushPeriod	Optional. This value sets the time period after which unmodified cache entries are flushed (value in seconds). Valid range: 0-100.

Return Values

None.

pub.sap.transaction:storeConfig

Returns the current configuration and cache parameter of the transaction store.

Input Parameters

None.

Return values

storeType	“FileSystem" for file based transaction store.
writeCache	“On” or “Off” depending on state of the cache.
cacheSize	Number of entries currently held in the transaction cache.
cacheTimeToLive	Current maximum time-to-live period for a cache entry (in seconds).
cacheFlushPeriod	Current period after which unmodified cache entries are flushed (in seconds).

pub.sap.transaction:delete

Deletes one transaction from the transaction store.

Input Parameters

$tid	The TID of the transaction

Return values

None.

pub.sap.transaction:deleteAll

Deletes the entire transaction store. Exercise care if you choose to use this.

Input Parameters

None.

Return values

deletedEntries	Total number of deleted transactions that matched the filter criteria.

pub.sap.transaction:sweep

This service can be used to cleanup the transaction store that matches the given filter criteria. For example, set it up in the Scheduler as follows:

1 Create a Flow Service, which invokes pub.sap.transaction:sweep.

2 In the Flow Service, define the inputs using `Set Value`.

3 Create a scheduler job via **Server ▸ Scheduler** that executes the Flow Service.

Input Parameters

elapsedTime	Filter criteria: time in minutes since the transaction has been modified the last time.
maxTrxCount	Optional. Maximum number of transaction to be deleted in one execution of the Service.
sender	Optional. Filter criteria: name of the sender of the transaction.
receiver	Optional. Filter criteria: name of the receiver of the transaction.
msgType	Optional. Filter criteria: name of the message type of the transaction.
state	Optional. Filter criteria: state of the transaction. Valid values are: ■ Created ■ Executed ■ Committed ■ Rolledback ■ Confirmed

Return Values

deletedEntries	Total number of deleted transactions that matched the filter criteria.

Transport Services

pub.sap.transport.ALE:InboundProcess

Receives an IDoc and forwards it to the routing listener.

Input Parameters

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
$tid	Optional. The TID of the transaction.
$action	Optional. One of the following transaction codes:

Code	Meaning
1	Execute (default)
4	Confirm

Return Values

None.

pub.sap.transport.ALE:OutboundProcess

Sends an IDoc to an SAP system.

Important! If *$action* is set to 4, this service behaves the same as service pub.sap.client:confirmTID.

Input Parameters

transportParams	A document with the following key/value pair.

Key	Value
serverName	Alias of the SAP system to which you want to send the IDoc. The alias must match a configured RFC connection alias at the SAP Adapter.

iDocList	Optional. Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
$tid	Optional. The TID of the transaction.

$action — One of the following transaction codes:

Code	Meaning
1	Execute (default)
4	Confirm

$queueName — Optional. Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario.

Return Values

$runtime — Total invocation time (ms), including processing time within the SAP Adapter.

$rfctime — Time (ms) spent within the RFC library to complete the invocation.

$encoding — MIME-compliant character set corresponding to the session's SAP code page.

$call — Flag indicating whether pipeline represents a request (true) or a response (false) of a function module.

pub.sap.transport.ALE:getRoutingInfo_Default

Default service to retrieve routing information from a list of IDocs.

Note: This service can also handle IDOC_CONTROL and IDOC_DATA record lists (for IDocs version 3) or IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40 record lists (for IDocs version 4) as IDoc input parameters instead of iDocList.

Input Parameters

iDocList — Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.

Return Values

iDocList — Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.

sender — Retrieved from sender partner number from the header of the first IDoc.

receiver — Retrieved from receiver partner number from the header of the first IDoc.

msgType — Retrieved from message type from the header of the first IDoc.

pub.sap.transport.IS:InboundProcess

Receives a message from a remote Integration Server and forwards it to the routing listener, which determines the routing notification to use to route the message based on the sender, receiver, and message type specified in the input parameters. After determining the routing notification to use, it invokes the flow service that processes the matching routing notification. This can be called with an arbitrary message to pass it on to the routing listener.

Important! It is recommended that you use this service as the receiving point when you are connecting two Integration Servers.

Input Parameters

In addition to the parameters specified below, the input parameters must also include the message to be routed through the SAP Adapter. Use a key for the message that the outbound transport expects.

sender	The sender of a message.
receiver	The receiver of the message.
msgType	Message type of the message.
$tid	Optional. The TID of the transaction.
$action	Optional. One of the following transaction codes:

Code	Meaning
1	Execute (default)
4	Confirm

Return Values

Depends on transaction and invoked routing notification.

pub.sap.transport.IS:OutboundProcess

Invokes a service on a local or remote webMethods Integration Server.

Input Parameters

transportParams — A Document with the following key/value pairs.

Key	Value
folderName	Folder name for the service to invoke.
serviceName	Name of the service to invoke.
serverAlias	Optional. Alias of the webMethods Integration Server on which the service is to be invoked. To invoke a service on a remote server: ■ Specify the alias for the remote server. The default value is local, which indicates the local server. Use local to invoke a service on the local server that is not password protected. ■ To invoke a password protected service on the local server, set up an alias for the local server, specifying a user name and password that has access to the password protected service. Then, specify that alias name.
valueScope	Optional. The scope for the session. SESSION indicates the connection is to be saved in your own session. GLOBAL indicates the connection is to be saved in a shared area. Use SESSION when the work being performed requires state be maintained.

data — Pass the message data to this record. If data is not set, the complete pipeline will be passed to the receiving service.

$tid — Optional. The transaction ID. Specify a transaction ID if the document should be sent as a transaction.

$action Optional. One of the following transaction codes:

Code	Meaning
1	Execute (default)
4	Confirm

Return Values

The pipeline contains the result of the service that was invoked.

pub.sap. transport.RFC:InboundProcess

Receives an inbound RFC or an RFC-XML document that is to be routed through the SAP Adapter.

Input Parameters

This service requires the following input parameters:

sbcHeader Values object that represents the key/value pairs in the SBCHEADER table. For information about what to include in the array of records, refer to "The SBCHEADER Table" on page 138.

$rfcname Optional. The name of the function module.

$tid Optional. The TID of the transaction.

If you want to process the RFC call asynchronously using tRFC, specify the transaction ID. If you would like to perform a synchronous RFC call, do not specify this parameter.

$action Optional. The transaction state. Specify one of the following codes:

Code	Meaning
1	Execute (default)
4	Confirm

Return Values

Depends on transaction and invoked routing notification.

pub.sap.transport.RFC:OutboundProcess

Invokes an RFC on an SAP system. This service can handle both RFC and tRFC calls.

Input Parameters

This service requires the following input parameters:

transportParams A document with the following key/value pair.

Key	Description
serverName	The SAP system alias for the SAP system on which invoke the RFC. The alias must match a configured RFC connection alias at the SAP Adapter.

$tid Optional. The TID of the transaction.

If you want to process the RFC call asynchronously using tRFC, specify the transaction ID. If you would like to perform a synchronous RFC call, do not specify this parameter.

$action Optional. The transaction state. Specify one of the following codes:

Code	Meaning
`1`	Execute (default)
`4`	Confirm

$rfcname Optional. RFC that should be called

$queueName Optional. Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario.

Return Values

The pipeline contains the result of the invoked function module.

$runtime Total invocation time (ms), including processing time within the SAP Adapter.

$rfctime Time (ms) spent within the RFC library to complete the invocation.

$encoding MIME-compliant character set corresponding to the session's SAP code page.

$call Flag indicating whether pipeline represents a request (true) or a response (false) of a function module.

pub.sap.transport.XML:InboundProcess

Can be called (e.g. using HTTP post) with arbitrary XML documents to pass them on to the routing listener.

Input Parameters

node
$tid
: XML Node object that can be generated by calling pub.xml:xmlStringToXMLNode and is automatically generated when posting an XML document via HTTP (using text/xml).

  Optional. The TID of the transaction.

$action
: Optional. The transaction state. Specify one of the following codes:

Code	Meaning
1	Execute (default)
4	Confirm

Return Values

Depends on transaction and invoked routing notification.

pub.sap.transport.XML:OutboundProcess

Use this service to post an XML document to a specified URL.

This transport service includes pub.client:http that performs HTTP POST. The XML transport is basically used for posting XML documents to external Web servers. If you want to communicate with another webMethods' Integration Server, you can use pub.remote:invoke.

Input Parameters

This service requires the following input parameters from the Values object.

transportParams
: A document with the following key/value pair.

Key	Description
url	The URL to which to post the document.

	xmlType	The XML dialect to use. Specify `Values-XML` if you want the XML in webMethods native XML format. ■ Specify `SAP-XML` if you want the XML in a format that is compliant with the SAP XML specification. For an IDoc, IDoc-XML is used; for an RFC, RFCXML (XRFC) is used. Note that the IDoc must be available as input to this service. ■ `Arbitrary XML` allows to post any XML document. ■ `SOAP XRFC` can be selected as additional XML dialect. This equals to XRFC (RFC-XML) with a SOAP envelope (higher than SOAP 1.1).
	useTextXml	Optional. Flag that overwrites the content-type of bXML, XRFC and IDoc-XML to text/xml, if set to true.
	useUTF8	Optional. Flag that allows to force the XML rendering engines to use UTF-8 as encoding for the resulting documents.
	httpUser	Optional. The user name to supply for a user name/password authentication challenge.
	httpPassword	Optional. The password to supply along with httpUser for a user name/password authentication challenge.
	useBAPI	`ON` or `OFF`. If set to `ON`, the pipeline is encoded using bXML for Business Objects.
	objectName	The name of the business object, to which the call should be mapped.
	bapiName	The name of the BAPI method, to which the call should be mapped
xmlData		Optional. An XML document as string, only used if transportParams/xmlType is `Arbitrary XML`.
$encoding		Optional. The encoding used by the XML request document.
$tid		Optional. The transaction ID. Specify a transaction ID if the document should be sent as a transaction.

$action Optional. The transaction state. Specify one of the following codes:

Code	Meaning
1	Execute (default)
4	Confirm

Return Values

The return values depend on the document that is posted. The return value is a Values object representation of the answer to the posted document.

pub.sap.transport.BAPI:InboundProcess

When posting bXML documents representing a BAPI of a Business Object to this service, it will take care of passing the document to the routing listener.

Input Parameters

sbcHeader
objectName Document that contains the key/value pairs with routing information. Will be generated by the bXML parser.

bapiName Optional. Name of the business object that is represented in the pipeline.

Optional. Name of the BAPI that is represented in the pipeline.

$action Optional. The transaction state. Specify one of the following codes:

Code	Meaning
1	Execute (default)
4	Confirm

$tid Optional. The TID of the transaction.

Return Values

Depends on transaction and invoked routing notification.

pub.sap.transport.BAPI:OutboundProcess

Executes a BAPI in an SAP system. This transport service can handle both asynchronous and synchronous execution of a BAPI.

Input Parameters

transportParams A document with the following key/value pair.

Key	Description
serverName	The SAP system alias on which to execute the BAPI. The alias must match a configured RFC connection alias at the SAP Adapter.
mode	Allows to restrict the way how the BAPI can be invoked:

Value	Meaning
`no restrictions`	Will be executed as the client requested
`synchronous only`	Only synchronous call is allowed, if client passes a *$tid* (indicating an asynchronous call) an exception is thrown.
`asynchronous only`	Only asynchronous call is allowed, if client does not pass a *$tid* (indicating a synchronous call) an exception is thrown.

$tid Optional. The TID of the transaction.

$action Optional. The transaction state. Specify one of the following codes:.

Code	Means
1	Execute (default)
4	Confirm

$queueName Optional. Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario.

objectName Optional. The name of the business object.

bapiName Optional. The name of the BAPI method.

Return Values

A pipeline that represents the response or an exception of the executed BAPI.

$runtime	Total invocation time (ms), including processing time within the SAP Adapter.
$rfctime	Time (ms) spent within the RFC library to complete the invocation.
$encoding	MIME-compliant character set corresponding to the session's SAP code page.
$call	Flag indicating whether pipeline represents a request (true) or a response (false) of a function module.

Specifications

pub.sap.listener:listenerAuthorizationCheck.

To provide application specific authorization handling with SNC enabled RFC servers (listeners), the user might provide a service that implements this specification. This service will be called by the RFC server (listener) to check for authorization.

This service has to return "Granted" in the "access" field if the request should be accepted. If no service is specified access will be granted. If an exception happens during execution of this service or a different string than "Granted" will be returned the access will be denied.

Input Parameters

functionName	The function that has been called by a remote client
partnerName	The partner (system) name where the request comes from
key	The authorization key as binary data depending on the mode
mode	The mode of the authorization: ("SNC" = Secure Network compliant authorization, "Basic" = Basic authorization). Currently, mode will always be Secure Network compliant authorization ("SNC").

Return Values

access	String. Should be string "granted" if the request should be accepted.

pub.sap.transaport.ALE:aleRoutingInfo_Default

To allow content based routing for all IDocs, the user might provide a service that implements this specification. This service will be called prior to selecting the right routing notification by the routing listener.

Input Parameters

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.

Return Values

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
sender	Contains the sender value as determined by this ALE default routing info service.
receiver	Contains the receiver value as determined by this ALE default routing info service.
msgType	Contains the message type value as determined by this ALE default routing info service.

pub.sap.transaport.ALE:aleRoutingInfo

To allow message type specific content based routing, the user might provide a service that implements this specification. This service will be called prior to selecting the right routing notification by the routing listener but after the default ALE routing info service was called.

Input Parameters

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
sender	Contains the sender value as determined by this ALE default routing info service.
receiver	Contains the receiver value as determined by this ALE default routing info service.
msgType	Contains the message type value as determined by this ALE default routing info service.

Return Values

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
sender	Contains the sender value as determined by this ALE default routing info service.
receiver	Contains the receiver value as determined by this ALE default routing info service.
msgType	Contains the message type value as determined by this ALE default routing info service.

pub.sap.transaport.ALE:aleMappingInfo_Default

To allow mapping for all IDocs, the user might provide a service that implements this specification. This service will be called prior to sending an IDoc to an SAP system using the pub.sap.transport.ALE:OutboundProcess service.

Input Parameters

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.

Return Values

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
msgType	Contains the message type value as determined by this ALE default routing info service.

pub.sap.transaport.ALE:aleMappingInfo

To allow message type specific mapping for IDocs, the user might provide a service that implements this specification. This service will be called prior to sending an IDoc to an SAP system using the pub.sap.transport.ALE:OutboundProcess service but after the default ALE mapping info service was called.

Input Parameters

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.
msgType	Contains the message type value as determined by this ALE default routing info service.

Return Values

iDocList	Contains the IDoc(s) as an object of the type com.sap.mw.idoc.IDoc.DocumentList.

pub.sap.transaport.XML:xmlRoutingInfo

To allow routing for incoming XML documents, the user must provide a service that implements this specification. This service will be called prior to selecting the right routing notification by the routing listener.

Input Parameters

node	XML Node object that can be generated by calling pub.xml:xmlStringToXMLNode and is automatically generated when posting an XML document via HTTP (using text/xml).
- OR -	
document	XML in hierarchical structure.

Return Values

document	XML in hierarchical structure.
sender	Contains the sender value as determined by this XML routing info service.
receiver	Contains the receiver value as determined by this XML routing info service.
msgType	Contains the message type value as determined by this XML routing info service.
$tid	The TID of the transaction as determined by this XML routing info service.
$action	Optional. The transaction state as determined by this XML routing info service. One of the following codes:

Code	Meaning
1	Execute (default)
4	Confirm

Sample Services

sample.sap:handleIDocXMLPost

Submit an IDoc-XML message to ALE InboundProcess on the SAP Adapter.

Input Parameters

xmlData String. The XML message of the IDoc document to be sent. Required.

$action Optional. The transaction state. Specify one of the following codes:

Code	Meaning
1	Execute (default)
4	Confirm

Return Values

None.

sample.sap:handleRfcXMLPost

Submit an RFC-XML or bXML message to an SAP system to invoke the associated RFC function.

Input Parameters

serverName String The alias for the SAP system on which to execute the RFC. The alias must match a configured RFC connection alias at the SAP Adapter.

xmlData String The XML message of the RFC-XML document.

$envelope Optional. bXML or RFC-XML, document type used for the post.

Return Values

xmlData XML message representing the response of the function module.

Note: This service is designed only for executing function modules that do not require an explicit commit.

sample.sap:handlebXMLPost

Submits a bXML message to an SAP system to invoke the associated BAPI.

Input Parameters

serverName Alias for the SAP system on which to execute the BAPI. The alias must match a configured RFC connection alias at the SAP Adapter.

xmlData String XML message representing a BAPI request in bXML.

mode Allows you to choose the way how the BAPI will be invoked:

Value	Meaning
`sync`	synchronous execution
`async`	asynchronous execution
`routing`	apply routing matching routing info in the document

Return Values

xmlData XML document representing the response of the BAPI.

Note: This service is designed only for executing BAPIs that do not require an explicit commit.

sample.sap.Helpers:writeSAPXMLFile

This service will convert RFCs or IDocs coming from an SAP system to SAP-XML (IDoc-XML or RFC-XML) and write the result to an output file in the WmSAP/pub directory. To call it, create a routing notification, and assign it this service.

Input Parameters

None.

Return Values

None.

sample.sap.Helpers:invokeBapiReturningBXml

This internal service is invoked by handleBXmlPost.

Input Parameters

serverName Alias for the SAP system on which to execute the BAPI. The alias must match a configured RFC connection alias at the SAP Adapter.

mode Allows you to choose the way how the BAPI will be invoked:

Value	Meaning
`sync`	synchronous execution
`async`	asynchronous execution
`routing`	apply routing matching routing info in the document

Return Values

xmlData XML message representing the response of the BAPI.

sample.sap.Helpers:invokeAndReturnXml

This internal service is invoked by handleRfcXmlPost.

Input Parameters

serverName	The SAP connection alias for the SAP server on which to execute the function module. The alias must match a configured SAP connection on the SAP Adapter.
$rfcName	Name of the function module to be invoked.
$tid	Optional. Specify TID for transactional invoke.
$envelope	Optional. Set to "bXML" to encode the results of the function module call in bXML format.

Return Values

xmlData	XML message representing the response of the function module.

sample.sap.Helpers:encodeRPCResponse

Sample service used to encode a Document (<key/value> pairs) as RPC-XML message.

Input Parameters

document	Document containing key/value pairs.

Return Values

xmlData	RPC-XML string representing the input document.

sample.sap.idoc.Mappings:orders

Example service for content-based routing of ORDERS IDocs. The service determines routing information (sender, receiver, msgType) for the given IDoc of type ORDERS.

For the description of input parameters and return values, please see service specification pub.sap.transaport.ALE:aleRoutingInfo.

sample.sap.idoc.Mappings:ordrsp

Example service for mapping ORDRSP IDocs. The service does some mapping of fields for the given ORDRSP IDoc.

For the description of input parameters and return values, please see service specification pub.sap.transaport.ALE:aleMappingInfo.

SAP Listener Services

pub.sap.listener:getAttributes

Returns information specific to one RFC server connection session:

Input Parameters

None.

Return Values

Document named *rfcAttributes* containing specific information about one RFC connection session:

destination	String. Destination of the connection.
ownHost	Hostname (or IP) of the machine the Integration Server is running on
partnerHost	Hostname (or IP) of the machine the SAP system is running on
systemNumber	System number of the SAP system
systemID	Unique three-letter-ID of SAP system (in local network)
client	Client the session is connected to
user	SAP user that has connected with this session
language	Logon language
ISOLanguage	ISO name for logon language
ownCodepage	SAP codepage this connection is using
partnerCodepage	SAP codepage the SAP system is running on
ownRelease	Version of the loaded RFC library used by the SAP Adapter
partnerRelease	Release of the SAP system
kernelRelease	Kernel release of the SAP system

partnerType	RFC type of the partner; R/3 (3), R/2(2) or external RFC server (E)
trace	Flag indicating whether trace is turned on(true) or off(false) for this connection
ownType	RFC type of the SAP Adapter; should always be E
rfcRole	Role of the SAP Adapter in this call; should always be: C (for client)
CPICConversationID	CPIC ID of the connection (low level protocol information)
encoding	IANA-encoding that is equivalent to the SAP codepage used, e.g. ISO-8859-1
charset	Charset that is equivalent to the SAP codepage used, e.g. ISO8859_1 (used in Java constructors)
bytesPerChar	Number of maximum bytes used per char in the codepage

webMethods SAP Adapter IDoc Java API

See *webMethods_directory*\IntegrationServer\WmSAP\pub\doc\api\index.html.

webMethods

APPENDIX E

Deprecated Services

List of Deprecated Services

In the following all services and specifications are listed that have been replaced by new ones as of release 4.6. The old services are still available, but should be no longer used.

Previous	New service/specification
pub.sap.idoc:transformFlatToHierarchy	pub.sap.idoc:IDocToDocument
pub.sap.idoc:transformHierarchyToFlat	pub.sap.idoc:documentToIDoc
pub.sap.idoc.routing:registerService	Use the SAP AdapterAdministration UI instead.
pub.sap.idoc.routing:unregisterService	Use the SAP AdapterAdministration UI instead.
pub.sap.idoc:idocToString	pub.sap.idoc:encodeString
pub.sap.idoc:iDocToRecord	pub.sap.idoc:iDocToDocument
pub.sap.idoc:recordToIDoc	pub.sap.idoc:documentToIDoc
pub.sap.idoc.routing:inbound	pub.sap.transport.ALE:aleRoutingInfo
pub.sap.idoc.routing:inboundDefault	pub.sap.transport.ALE:aleRoutingInfo_Default
pub.sap.idoc.routing:outbound	pub.sap.transport.ALE:aleMappingInfo
pub.sap.idoc.routing:outboundDefault	pub.sap.transport.ALE:aleMappingInfo_Default
wm.PartnerMgr.gateway.transport.B2B:InboundProcess	pub.sap.transport.IS:InboundProcess
wm.PartnerMgr.gateway.transport.B2B:OutboundProcess	You should directly select the service to call at the routing notification.
wm.PartnerMgr.gateway.transport.FTPTransport:OutboundProcess	You should directly select the service to call at the routing notification.
wm.PartnerMgr.gateway.transport.EmailTransport:OutboundProcess	You should directly select the service to call at the routing notification.

webMethods

APPENDIX F

Working with Code Pages

Using Different Code Pages

Use the following procedure if you handle data that is encoded (or that you want to encode and send out) in a different coding from your operating systems default code page.

To Receive Data from HTTP, FTP, E-mail or File

- You can use the Services pub.client:http, pub.client:ftp, pub.client:smtp and pub.file:getFile to load a multi-byte document into the pipeline. Always use the option `loadAs=bytes`.
- Convert the binary data using the Services pub.string:bytesToString or a combination of pub.xml:xmlStringToXMLNode (also accepts bytes) and pub.xml:xmlNodeToDocument.
- Set the input parameter encoding to the mime Encoding in which the bytes have been encoded, (one that is supported by the functions `sun.io.ByteToChar`*XXX*`.class`, where *XXX* stands for the encoding) for example: ASCII, ISO2022 or SJIS.

To Send Data Via HTTP, FTP, E-mail or Save It to a File

- Ensure that you do not pass any String objects into the Services pub.client:http, pub.client:ftp, pub.client:smtp or save a String into a file.
- Call pub.string:stringToBytes with the correct encoding parameter.
- Pass the bytes into ftp/http or into your Service which writes the file. (This service should use `java.io.FileOutputStream` to write the file, not FileWriter.) That way the message is sent out of SAP Adapter with proper encoding.

To Encode Data from SAP Systems

Use the following procedure if you are using pub.sap.rfc:encode

To create the standard SAP XML you would use the service pub.sap.rfc:encode, which handles encoding automatically.

Note: If you wish to create other XML-documents from SAP data, you should be careful to avoid problems with code pages whenever you receive data from an SAP system (e.g. a response to an outbound Remote Function Call or via an inbound call).

You need to set the encoding attribute in the XML-document manually:

Use the following steps if you are using pub.xml:documentToXMLString

1 Extend your document by adding the attribute `@encoding` as a String as a child of the root element.

2 Insert the proper default value (e.g. Shift-JIS, if your source data is Shift-JIS encoded).

3 In the pub.xml:documentToXMLString Service, map your document to input field "document".

webMethods

APPENDIX G

Using BizTalk Envelopes with the SAP Adapter

Overview

The SAP Adapter uses the BizTalk XML envelope as the transport envelope for the transmission of XML business documents. The BizTalk envelope is implemented on the SAP Adapter in accordance with the *BizTalk Framework 1.0a Independent Document Specification of January, 7th 2000*.

As BizTalk specifications are open to implementation specific interpretations of some of the provided XML elements, this chapter describes how the SAP Adapter copes with BizTalk envelope tags that are not specified clearly enough. BizTalk strongly differentiates between a transport header and an application specific body, which results in the following overall structure of the BizTalk envelope:

```
<biztalk_1 xmlns="urn:schemas-biztalk-org:BizTalk/biztalk-1.0.xml">
   <header>
      <!-- Header and processing information is contained here -->
   </header>
   <body>
      <!-- Business transaction information is contained here -->
   </body>
</biztalk_1>
```

The BizTalk document header information is contained within the <header> element. This header contains information used for handling and processing the document. The business document passed in the message is contained within the <body> element. For more details concerning the standard BizTalk envelope refer to the *BizTalk Framework 1.0a Independent Document Specification*.

Use of the BizTalk Header

Introduction to Standard BizTalk Header Fields

BizTalk defines some standard XML elements for using with its envelope. The most important are:

- The <message> element with its sub-elements <messageID>, <sent>, <subject> can be used to identify a message.
- The <to> and <from> elements can be used to identify the sender and receiver in a communication process. Both may contain the sub-elements <address> and <state>.
- The <manifest> element can be used to describe the business documents delivered in the body.

A full BizTalk XML header may look like this (taken from the Microsoft BizTalk specification):

```
<?xml version='1.0' ?>
<biztalk_1 xmlns="urn:schemas-biztalk-org:BizTalk/biztalk-1.0.xml">
   <header>
      <delivery>
         <message>
            <messageID>xyzzy:8</messageID>
            <sent>1999-01-02T19:00:01+02:00</sent>
            <subject>Purchase Order</subject>
         </message>
         <to>
            <address>http://www.fabrikam.com/recv.asp</address>
            <state>
               <referenceID/>
               <handle/>
               <process/>
            </state>
         </to>
         <from>
            <address>mailto:foo@contoso.com</address>
            <state>
               <referenceID>123</referenceID>
               <handle>7</handle>
               <process>myprocess</process>
            </state>
         </from>
      </delivery>
      <manifest>
         <document>
            <name>PO</name>
            <description>Purchase Order</description>
         </document>
      </manifest>
   </header>
   <!-- body definition here -->
<biztalk_1>
```

For further details of the BizTalk XML envelope, refer to the *BizTalk Framework 1.0a Independent Document Specification*.

The SAP Adapter only uses a subset of these available header elements, which is described in the following chapters.

Representation of Routing and Address Information

The SAP Adapter needs to identify the sender and receiver of an XML message. Usually, these are logical systems, as defined inside the SAP system.

This information is transported by the BizTalk header element `<delivery>`. Within this element, the two sub-elements `<to>` and `<from>` can be defined.

These elements are used to exchange the routing information needed by the SAP Adapter. The names of the partners are put into the sub-element `<address>` of the `<to>` or `<from>` element.

BizTalk requires an URI as the content of the address element. The names of the SAP logical systems are encoded as a Universal Resource Name (URN) by putting the prefix `urn:sap-com:logical-system:` in front of the logical system identifier. For example: The logical system SAPCLNT001 would be represented as `urn:sap-com:logicalsystem:SAPCLNT001`.

Note: Although these identifiers are expressed as URNs, they are not globally unique; the names of logical systems can be setup freely for each SAP System domain.

This results in an XML document like the following:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<biztalk_1 xmlns="urn:schemas-biztalk-org:BizTalk/biztalk-1.0.xml">
   <header>
      <delivery>
         <message>
            <messageID>0A125F1315B3A11B00000035</messageID>
            <sent>2000-06-07T09:22:40</sent>
         </message>
         <to>
            <address>urn:sap-com:logical-system:SAPCLNT001</address>
         </to>
         <from>
            <address>urn:sap-com:logical-system:SAPADA0001</address>
         </from>
      </delivery>
   </header>
   <!-- Body definition here -->
</biztalk_1>
```

Representation of SAP Transactions

When using tRFC or IDocs, you have to specify a unique SAP transaction ID for each call to ensure delivery. This transaction ID can be either resolved from an SAP system as described in this guide or built locally. It should always consist of 24 hexadecimal characters and be globally unique.

Because BizTalk does not support transaction IDs (TIDs), the SAP Adapter uses the `<referenceID>` element, which is a sub element of the `<state>` element.

Note: The `<state>` element can also be a sub-element of the `<to>` or `<from>` element.

Transaction IDs must be put in the `<referenceID>` element that belongs to the receiver's address information.

For example:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<biztalk_1 xmlns="urn:schemas-biztalk-org:BizTalk/biztalk-1.0.xml">
   <header>
      <delivery>
         <message>
            <messageID>0A125F1315B3A11B00000035</messageID>
            <sent>2000-06-07T09:22:40</sent>
         </message>
         <to>
            <address>urn:sap-com:logical-system:SAPCLNT001</address>
            <state>
               <referenceID>120A135FB315A11B00003500<referenceID>
            </state>
         </to>
         <from>
            <address>urn:sap-com:logical-system:SAPADA0001</address>
         </from>
      </delivery>
   </header>
   <!-- Body definition here -->
</biztalk_1>
```

Further BizTalk Header Fields

BizTalk requires the use of the XML element `<message>` to identify a single message. It defines the following sub-elements:

XML Element	Description
`<messageID>`	A unique identifier, generated for each exchanged document. The SAP Adapter will always generate a new message ID for both request and response messages.
`<sent>`	Timestamp of the message creation. When using BAPIs with ALE, this timestamp is used to build the serialization ID of the IDoc control block. The element must contain date and time formatted according to ISO 8601 (first edition June 15, 1988). The format to use is Calendar date and local time of the day. The syntax is: `CCYY-MM-DDThh:mm:ss.` For example: `2000-06-19T18:59:02` describes June 19, 2000 18 hours 59 minutes 2 seconds.
`<subject>`	May be used to specify an additional description for the message. **Important!** This element is not used by the SAP Adapter.

Important! BizTalk provides the possibility to exchange manifest information in the header. This feature is not used by the SAP Adapter.

Error Handling

BizTalk does not describe its own error handling concept in its current specification. However, it recommends that you define application-specific error documents to handle application errors. For server-related errors, it describes using standardized XML exception descriptors.

The SAP Adapter distinguishes between two major groups of errors:

- Errors in the XML processing and conversion layers of the SAP Adapter, and critical errors in the SAP systems that cause a termination of the connection.
- Application-specific errors such as errors that were foreseen by the application developers and therefore defined at the interface definition in the SAP system. For example: a receipt could not be processed because of business-level problems.

This section includes the following topics:

Representation of Communication and Processing Errors

Errors which are caused inside the SAP Adapter or by the technical layers in the SAP system are represented by a uniform XML fault-descriptor element. This fault element has been defined following the design principles used for SOAP and SAP XRFC error handling.

This fault element is transferred in the body of a BizTalk XML envelope and introduced by the fault-XML element (which is defined in the namespace `urn:sapcom:document:sap:business` by applying a specific prefix).

This fault element has the following sub-elements:

XML Element	Description
`<faultcode>`	A number specifying the class of error. Compatible with the Microsoft SOAP XML framework specification. The SAP Adapter always sets this value to 401 to indicate an application-specific error.
`<faultstring>`	Internal SAP Adapter code for this exception.
`<detail>`	Details of the specified sub-elements `<name>` and `<message>`
`<name>`	Name of an exception to use with the SAP system as the ABAP exception identifier.
`<message>`	Error message with the specified sub-element `<text>`. (Optional; this is not always fully specified).
`<text>`	The text of the error message.

An example of an exception document:

```
<?xml version="1.0" encoding="iso-8859-1"?>
   <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
      <header>
         <delivery>
            <message>
               <messageID>0A125F1315B39EDB00000013</messageID>
               <sent>2000-06-19T18:59:02</sent>
            </message>
            <to>
               <address>urn:sap-com:logical-system:SAPCLNT001</address>
            </to>
            <from>
               <address>urn:sap-com:logical-system:SAPADA0001</address>
            </from>
         </delivery>
      </header>
      <body>
         <sap:Fault xmlns:sap="urn:sap-com:document:sap:business"
                xmlns="">
            <faultcode>401</faultcode>
            <faultstring>
               com.wm.adapter.sap.error.SAPBasicException
            </faultstring>
            <detail>
               <name>SBC_EXCEPTION</name>
               <message>
                 <text>Business object named Bank2 does not exist in SAP
                       System U9C</text>
               </message>
            </detail>
         </sap:Fault>
      </body>
   </biztalk_1>
```

Representation of Application Errors

Application specific errors are always described as application specific XML business documents.

These exception documents are described in Appendix H, “Using IFR-XML Format with the SAP Adapter”. For SAP’s standard interfaces, the relevant exception documents are provided in the SAP interface repository as part of the XML schema for response documents.

Note: These response documents are always exchanged as business documents that are part of the BizTalk body.

webMethods

APPENDIX H

Using IFR-XML Format with the SAP Adapter

Overview

With the XML format defined in the Interface Repository, SAP establishes a standardized exchange of business management data on a semantic level across the web. XML documents which are generated according to the XML format and defined in the Interface Repository are called *business documents*.

The number of business documents exchanged between two partners to invoke a business management function depends on the underlying interface type and the usage scenario. During a synchronous call of a BAPI or RFC, two business documents (request and response document) are exchanged. During an asynchronous call, only a request document is exchanged.

Characteristics of business documents are:

- Use of XML envelopes
- Common structure for request and response documents
- Common way of representing parameters
- Common error handling concepts

Note: IDocs do not support all of these features

Use of XML Envelopes

Business documents are transmitted within an XML envelope. Due to the fact that they only contain business management data they can be transmitted via arbitrary XML standard envelopes (e.g. BizTalk, SOAP, …).

The SAP Adapter supports the transport of business documents within the BizTalk envelope. More details concerning the use of the BizTalk envelope for the transport of business documents can be found in Appendix G, “Using BizTalk Envelopes with the SAP Adapter”.

Common Structure of Request and Response Documents

To ensure a standardized exchange of business management data the overall structure of request and response documents is the same regardless of which interface type is called. Generally three different types of business documents can be distinguished:

- Request business documents that are identified by a root element that has the same name as the corresponding interface.

  Example:

```
<doc:InterfaceName xmlns:doc="urn:…">
   <!-- request specific data -->
      …
</doc:InterfaceName>
```

- Response business documents that are sent back from the server if the request has been correctly executed. They are identified by a root element that has the concatenation of the interface name and the suffix `.Response` as its name.

  Example:

```
<doc:InterfaceName.Response xmlns:doc="urn:…">
   <!-- response specific data -->
      …
</doc:InterfaceName.Response>
```

- Exception business documents that are sent back from the server if application errors occurred during the execution of the request. They are identified by a root element that carries the concatenation of the interface name and the suffix `.Exception` as its name.

  Example:

```
<doc:InterfaceName.Exception xmlns:doc="urn:…">
   <!-- exception specific data -->
      …
</doc:InterfaceName.Exception>
```

Common Way of Representing Parameters

Parameters are generally represented as child elements of the business document root element. What parameter names are used as element names within the business document depends on the underlying interface type:

- In business documents for RFCs, the names of the parameters of the function module interface are used in the XML document.
- Business documents for BAPIs contain the parameter names as they are defined in the business object Repository (BOR).

The SAP-internal data structures of the parameters are displayed in a serialized form within the parameter element in accordance with the SAP specification *Serialization of ABAP Data in XML* (for further information, please refer to "Related Documentation" on page 13).

Common Error Handling Concepts

The different error handling concepts of BAPIs and RFCs are presented at the XML level in a uniform way.

Generally, exception business documents are structured as serialized ABAP OOExceptions which are passed within the root element (`InterfaceName.Exception`)of the exception document. These serialized exceptions are flexible enough to contain different kinds of error information. Consequently, both BAPI return parameters and function module exceptions are mapped to the representation of serialized exception objects.

The serialized ABAP-OO exception consists of the following XML elements:

XML Element	Description
`<Name>`	Identifies the exception. Normally a global name.
`<Message>` (optional) sub-elements:	The `<Message>` element references an SAP message that describes the error situation.
`<ID>`	Message class
`<Number>`	Message number
`<Text>` (optional)	Error description
`<Attributes>` (optional) sub-elements:	The <Attributes> element contains additional application specific information:
`<Collection>` (optional)	Table with serialized exceptions that represent single error messages, if error collections have to be returned. Each serialized exception within the <Collection>-element has the same structure as the wrapper exception.
`<AttributenameX>` `(multiplicity: 0..*)`	Named attributes (AttributenameX =Name of Attribute) in which additional application specific information can be returned

More detailed information about the XML format defined in the Interface Repository can be found at http://ifr.sap.com.

XML Format for BAPIs

For each BAPI, three different types of business documents exist:

- A request business document.
- A response business document (in case of a correct execution of the BAPI call).
- An exception business document (in case of an incorrect execution of the BAPI call).

The following sections describe these three types of business documents in more detail.

Structure of Request Business Documents for BAPIs

This is the general structure of request business documents for BAPIs:

```
<doc:BusinessObjectName.MethodName
     xmlns:doc="urn:sap-com:document:sap:business"
     [Key1=”…”, Key2=”…”, …]>
   <BOR_Parameter1>...</BOR_Parameter1>
          ...
</doc:BusinessObjektName.MethodenName>
```

Additionally, consider the following:

- The arrangement of BAPIs according to a business object is directly reflected by the element names of the business document. The root element that identifies the Request business document carries the concatenation of the business object name and the BOR method name as its name. The root element has the following characteristics:
  - It contains the namespace reference of business documents that refer to BAPIs (`"urn:sap-com:document:sap:business"`).
  - If the BAPI is an instance method, the root element contains an attribute for each key field of the corresponding business object that is named like the key field.

  Note: If the business object has more than one key field, the corresponding attributes should appear in the same sequence as the key fields defined in the BOR.

- Import parameters of the BAPI appear as sub-elements of the root element. These sub-elements are named like the parameters as they are defined in the BOR. Within these elements, the SAP-internal data structures are represented in a serialized form in accordance with the specification *Serialization of ABAP Data in XML* (for further information, please refer to “Related Documentation” on page 13).

Structure of Response Business Documents for BAPIs

A response business document is returned to the sender if the BAPI call could be processed without application errors. This means that the BAPI-return parameter only contains messages of the type *S*, *I*, or *W*. In that case, the Response business document contains the export parameters of the BAPI call.

This is the structure of response business documents for BAPIs:

```
<doc:BusinessObjektName.MethodenName.Response
    xmlns:doc="urn:sap-com:document:sap:business"
    [Key1=”…”, Key2=”…”, …]>
   <BOR_Parameter1>...</BOR_Parameter1>
         ...
</doc:BusinessObjektName.MethodenName.Response>
```

Additionally, consider the following:

- The arrangement of BAPIs according to a business object is directly reflected by the element names of the business document. The root element that identifies the request business document carries the concatenation of the business object name, the BOR method name and the suffix .Response as its name. The root element has the following characteristics:
  - It contains the namespace reference of business documents that refer to BAPIs (`“urn:sap-com:document:sap:business”`).
  - If the BAPI is an instance method or a factory method, the root element contains an attribute for each key field of the corresponding business object that has the same name as the key field.

Important! If the business object has more than one key field, the corresponding attributes should appear in the same sequence as the key fields defined in the BOR.

- Export parameters of the BAPI appear as sub-elements of the root element. These sub-elements have the same name as the parameters defined in the BOR. Within these elements, the SAP-internal data structures are presented in a serialized form in accordance with the specification *Serialization of ABAP Data in XML* (for further information, please refer to “Related Documentation” on page 13).
- If the BAPI is a factory method, possibly only the key fields of the created instance will be returned to the client. In this case, the response business document only consists of attributes.

Structure of Exception Business Documents for BAPIs

An exception business document is returned to the sender if the BAPI call could not be processed without application errors. This means that the BAPI return parameter contains at least one message of type *E* (error) or *A* (abort). In this case, the exception business document contains only the error descriptions.

These error descriptions have the following characteristics:

- Error messages of the BAPI return parameter are displayed as serialized exception objects on XML level. So the representation of the error description in the exception business document differs from the original BAPI return structure. Messages of type *E* are represented as exceptions with name *BapiError*. Messages of type *A* are represented as exceptions with the name *BapiAbort*.
- Status messages (messages of the type *S*, *I*, or *W*) that are also part of the Return parameter are still returned as the return parameter structure.

Important! All status messages will be mapped to the BAPIRET2 structure.

- The export parameters of the BAPI are not taken into account.

The detailed structure of the Exception business documents depends on whether the return parameter is a structure or a table.

Exception Document for Return Structures

If the return parameter is a structure, the exception business document contains one single exception that is created from the information of the return message. The exception is named *BapiError* or *BapiAbort* depending on whether the return message is of type *E* or *A*.

The XML representation of a return structure consists of following sub-elements:

XML Element	Corresponding Information from the Return Parameter
`<Name>`	„BapiError“, if value of field TYPE is „E“. „BapiAbort“, if value of field TYPE is „A“.

XML Element	Corresponding Information from the Return Parameter
<Message> (0..1)	
<ID>	Value of the ID field.
<Number>	Value of the NUMBER field.
<Text>	Value of the MESSAGE field.
<Attributes> (optional) <MESSAGE_V1> (optional), <MESSAGE_V2> (optional), <MESSAGE_V3> (optional), <MESSAGE_V4> (optional) <LOG_NO> (optional) <LOG_MSG_NO> (optional) <PARAMETER> (optional) <ROW> (optional) <FIELD> (optional) <SYSTEM> (optional)	The <Attributes> element contains the information of BAPIRET2 as sub-elements that have the same names as the corresponding fields of the BAPIRET2 structure.

Exception Document for Return Tables

Return tables consist of one or more rows that have the same structure as a return structure. If the return table contains at least one message of type *E* (error) or *A* (abort), an exception business document is returned to the client that has the structure of a collection exception. This exception is named *BapiError* or *BapiAbort* depending on whether the most critical message in the return table is of type *E* or *A*.

The collection exception is identified by a standardized message which is similar for all return tables. The messages of the return table are displayed within the <Collection> element of the collection exception, which is a sub-element of the <Attributes> element.

Each single message of type *E* or *A* is displayed as an exception that is grouped within the <Collection> element in the form of a table. Single messages are displayed as exceptions in an exception table in accordance with the specification *Serialization of ABAP Data in XML* (for further information, please refer to "Related Documentation" on page 13). Consequently, each exception is encapsulated in an <item> element that contains the exception data.

If, despite the error messages, the return table contains additional messages of type *S*, *I*, or *W*, these messages are displayed in a table of row type `BAPIRET2` within a `<Status>` element, which is a sub-element of the `<Attributes>` element.

If a BAPI return table contains error messages, the following XML-elements comprise the resulting exception business document:

XML Element	Corresponding Information from the Return Parameter
`<Name>`	„BapiError“, if value of field TYPE is „E“. „BapiAbort“, if value of field TYPE is „A“.

`<Message>`
`(0..1)`
`<ID>`
`<Number>`
`<Text>`
	The three sub-elements contain a generic message that is used for all return tables (for example: “During the execution of the BAPI one or more errors occurred”).

`<Attributes>`
`<Collection>`
`<item>`
`<Name>…</Name>`
`<Message>…</Message>`
`<Attributes> (optional)`
`...`
`</Attributes>`
`</item>`
`<item> (optional)`
`...`
`</item>`
`...`
`</Collection>`
	XML representation of each error message within the return table. Within the `<item>` elements, the data of a single return message is displayed as a serialized exception object.

XML Element	Corresponding Information from the Return Parameter
<Status> (optional) <item> <ID> (optional) <TYPE> (optional) <NUMBER> (optional) <MESSAGE> (optional) <MESSAGE_V1> (optional) <MESSAGE_V2> (optional) <MESSAGE_V3> (optional) <MESSAGE_V4> (optional) <LOG_NO> (optional) <LOG_MSG_NO> (optional) <PARAMETER> (optional) <ROW> (optional) <FIELD> (optional) <SYSTEM> (optional) </item> <item> (optional) … </item> … </Status>	The <Status> element contains a serialized table with row type BAPIRET2. Within the <Status> element, all return messages of type *S*, *I*, or *W* that are also returned in the BAPI return parameter are displayed.

XML Format for RFCs

Just as for BAPIs, there are three different types of business documents for each RFC:

- A request business document.
- A response business document, if the RFC call has been successfully executed.
- An Exception business document, if the RFC call failed.

The following sections describe these three types of business documents in more detail.

Structure of Request Business Documents for RFCs

This is the general structure of request business documents for RFCs:

```
<doc:FunctionModuleName
        xmlns:doc="urn:sapcom:document:sap:business:rfc:functions">
   <FuMod_Parameter1>...</FuMod_Parameter1>
      ...
</doc:FunctionModuleName>
```

Additionally, consider the following:

- The root element that identifies the request business document has the same name as the corresponding function module. It contains the namespace reference of business documents that refer to RFCs (`"urn:sap-com:document:sap:business:rfc"`).
- Import parameters of the RFC appear as sub-elements of the root element. These sub-elements have the same name as the parameters as they are defined in the function module interface. Within these elements, the SAP-internal data structures are represented in a serialized form in accordance with the specification *Serialization of ABAP Data in XML* (for further information, please refer to "Related Documentation" on page 13).
- Because it is not possible to distinguish tables in the function module interface in import tables and export tables, all tables that should be taken into account have to be represented as sub-elements.

Structure of Response Business Documents for RFCs

If the RFC can be executed without getting an exception, a response business document is returned to the sender. It contains the export data of the function module.

This is the structure of response business documents for RFCs:

```
<doc:FunctionModuleName.Response
        xmlns:doc="urn:sapcom:document:sap:business:rfc:functions">
   <FuMod_Parameter1>...</FuMod_Parameter1>
      ...
</doc:FunctionModuleName.Response>
```

Additionally, consider the following:

- The name of the root element that identifies the response business document is defined as the concatenation of the name of the corresponding function module and the suffix `.Response`. It contains the namespace reference of business documents that refer to RFCs (`"urn:sap-com:document:sap:business:rfc"`).
- Export parameters and tables of the RFC appear as sub-elements of the root element. These sub-elements have the same name as the parameters defined in the function module interface. Within these elements, the SAP-internal data structures are presented in a serialized form in accordance with the specification `Serialization of ABAP Data in XML` (for further information, please refer to"Related Documentation" on page 13).

Structure of Exception Business Documents for BAPIs

If the function module cannot be executed, the execution ends with a function module exception and an exception business document is returned to the client. It describes only the error situation. In this case, export parameters of the function module are not included.

This is the structure of exception business documents for RFCs:

```
<doc:FunctionModuleName.Response
        xmlns:doc="urn:sap-com:document:sap:business:rfc">
   <!--Representation of the serialized exception-->
      ...
</doc:FunctionModuleName.Response>
```

The representation of the function module exception as a serialized exception consists of the following XML elements:

XML Element	Description
<Name>	Name of the exception as defined in the function module interface.
<Message> (0..1)	The <Message> element contains information from the SY fields. Could be empty if the mapping was performed on an external middleware system.
<ID>	SY-MSGID
<Number>	SY-MSGNO
<Text>	Message text, which is defined for ID and number. In ABAP, evaluated with „message id... into..“
<Attributes> (optional)	Contains list of SY fields, which represent the variables of an error message.
<V1> (optional)	SY-MSGV1
<V2> (optional)	SY-MSGV2
<V3> (optional)	SY-MSGV3
<V4> (optional)	SY-MSGV4

XML Format for IDocs

To create an asynchronous communication, IDocs are exchanged between two SAP systems. There are two ways to define concrete IDocs to communicate asynchronously:

- As of Release 4.0, generate an IDoc from an existing BAPI.
- Create IDocs manually.

Therefore, two different ways of representing IDocs in XML can be distinguished:

- If the IDoc was generated from an existing BAPI, the asynchronous communication is performed by using the BAPI interface. Consequently, the exchanged business documents are created from the corresponding BAPI. The IDoc itself is not visible at XML level.
- IDocs that were defined manually and not generated from BAPIs require a separate XML representation. This representation is described in detail below.

XML Format for Manually Defined IDocs

This is the structure of business documents for manually defined IDocs:

```
<doc:IdocTypeName>
   <IDOC BEGIN="1">
      <EDI_DC40 SEGMENT="1">...</EDI_DC40>
      <E1SEGMENT1 SEGMENT="1">...</E1SEGMENT1>
         ...
   </IDOC>
</doc:IDocTypeName>
```

Additionally, consider the following:

- The root element, which identifies the business document for an IDoc, has the same name as the corresponding IDoc type.
- The IDoc itself is encapsulated by the element `<IDOC BEGIN="1"> ...</IDOC>`.The element contains the attribute `BEGIN` with the fixed value “1”.
- The IDoc control record is defined through the element `<EDI_DC40 SEGMENT="1">...</ EDI_DC40>`.
- Each IDoc segment is represented as a separate element that has the same name as corresponding IDoc segment. SAP segments usually have the prefix `E1`. These elements contain the attribute `SEGMENT` with the fixed value “1”, which defines the beginning of the segment.

  The element name for the fields within one segment are given by the name of the corresponding field. An XML document for an IDoc may contain empty fields. These fields are displayed as elements with the form `<FieldName/>`.

- The hierarchical structure of IDocs is represented by the structure of the XML document.
- The IDoc control record is handled in the XML document as a common segment.

Index

D

E

F

G

I

R

S

T

U

V

W

X