

webMethods Adapter for SAP Installation and User's Guide

Version 10.1

October 2018

This document applies to webMethods Adapter for SAP 10.1 and to all subsequent releases.

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

Copyright © 2008-2023 Software AG, Darmstadt, Germany and/or Software AG USA, Inc., Reston, VA, USA, and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA Inc. and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software AG and/or its subsidiaries is located at <https://softwareag.com/licenses/>.

Use of this software is subject to adherence to Software AG's licensing conditions and terms. These terms are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software AG Products / Copyright and Trademark Notices of Software AG Products". These documents are part of the product documentation, located at <https://softwareag.com/licenses/> and/or in the root installation directory of the licensed product(s).

Document ID: ADAPTER-WSP-IUG-101-20230614

Table of Contents

About this Guide.....	9
Document Conventions.....	10
Online Information and Support.....	11
Data Protection.....	12
 1 Overview of webMethods Adapter for SAP.....	 13
What Is webMethods Adapter for SAP?.....	14
Functional Highlights.....	15
Architecture and Components.....	18
Adapter Package Management.....	24
Adapter Connections.....	24
Adapter Services.....	26
Adapter Listeners and Listener Notifications.....	26
Viewing the Adapter's Update Level.....	28
Controlling Pagination.....	28
 2 System Requirements.....	 29
Overview.....	30
CPU.....	30
Disk Space.....	30
Memory.....	32
Miscellaneous.....	35
 3 Installing, Upgrading, and Uninstalling webMethods Adapter for SAP 10.1.....	 37
Overview.....	38
Requirements.....	38
The Integration Server Home Directory.....	38
Installing Adapter for SAP.....	38
Upgrading to Adapter for SAP 10.1.....	41
Uninstalling Adapter for SAP 10.1.....	43
 4 Package Management.....	 45
Overview.....	46
Managing the Adapter Package.....	46
Controlling Group Access.....	50
Using the Adapter in a Clustered Environment.....	50
 5 Adapter Connections.....	 57
Overview.....	58
Before Configuring or Managing Adapter Connections.....	58
Configuring Adapter Connections.....	58
Setting Up the SAP System for SNC Connections.....	64

Dynamically Changing a Service's Connection at Run Time.....	65
Enabling Adapter Connections.....	66
Viewing Adapter Connections.....	66
Editing Adapter Connections.....	68
Copying Adapter Connections.....	68
Deleting Adapter Connections.....	69
Disabling Adapter Connections.....	69
Testing the Execution of an RFC.....	70
Testing the Execution of a BAPI Via XML.....	71
 6 Using Command Central to Manage Adapter for SAP.....	77
Configuration Types.....	78
Working with Adapter for SAP Configuration Types.....	78
 7 Adapter Services.....	81
Overview.....	82
Creating an Adapter Service that Executes an RFC.....	82
 8 Adapter Notifications.....	87
Overview.....	88
Creating an RFC Destination on an SAP System.....	91
Listeners.....	94
Listener Notifications.....	103
Examples.....	120
 9 Generating Document Types.....	131
Generating Document Types for RFC Structure.....	132
Generating Document Types for IDocs.....	132
 10 Routing Messages Through Adapter for SAP.....	139
Introduction.....	140
Overview.....	141
Routing Notifications.....	145
Sending an RFC from an SAP System to Adapter for SAP.....	150
Sending a BAPI from an SAP System to Adapter for SAP.....	155
Sending IDocs with ALE from an SAP System to Adapter for SAP.....	158
Routing RFCs Through Adapter for SAP.....	159
Routing BAPIs Through Adapter for SAP.....	160
Routing IDocs Through Adapter for SAP.....	165
Mapping IDocs to Other Formats.....	167
Content-Based Routing and Mapping for IDocs.....	172
Routing Arbitrary XML Documents Through the > Routing/Mapping.....	174
 11 Transaction Handling.....	177
Managing Transactions and the Transaction Store.....	178
Viewing Transactions.....	178
Deleting Transactions.....	180

Automatic Cleanup of the Transaction Stores.....	181
Configuration Parameters for the Transaction Manager.....	182
Using the ALE Monitoring Features Via Adapter for SAP.....	183
Shared Transaction Store.....	188
12 Coding Client Applications and Services.....	197
Overview.....	198
Invoking RFCs from Adapter for SAP.....	198
Receiving IDocs from an SAP System.....	199
Constructing an IDoc with the SAP Java IDoc Class Library.....	200
Transaction Features for HTTP and SAP-XML.....	202
Calling a BAPI Synchronously from SAP System.....	203
Calling a BAPI Asynchronously from an SAP System.....	206
13 Security.....	211
Adapter for SAP Configuration.....	212
User Authentication Between Adapter for SAP and an SAP System.....	212
Installing Adapter for SAP According to Your Security Policy.....	217
14 Managing the DDIC Cache.....	219
Data DICTIONary Cache (DDIC Cache).....	220
Viewing Information in the DDIC Cache.....	220
Removing Information from the DDIC Cache.....	222
15 Managing SAP User Store.....	225
Managing the SAP User Store.....	226
Adding Entries to SAP User Store.....	228
Changing Entries in SAP User Store.....	230
Removing Entries from SAP User Store.....	232
16 Predefined Health Indicator.....	235
Predefined Health Indicator.....	236
17 Administrator APIs.....	237
Administrator APIs.....	238
18 Configuration Variables Templates for Adapter Assets in Microservices Runtime.....	239
Configuration Variables Templates for Adapter Assets in Microservices Runtime.....	240
19 Logging and Monitoring.....	241
Overview.....	242
Logging.....	242
Monitoring Statistics for Each Adapter for SAP Connection.....	243
Monitoring Adapter for SAP Performance.....	245

A Package Contents.....	247
Package Layout.....	248
B Adapter Configuration.....	249
General Adapter for SAP Settings.....	250
server.cnf.....	250
C ABAP Types in Adapter for SAP.....	257
ABAP Types.....	258
D Built-in Services.....	259
SAP Client Services.....	260
XRFC Services.....	272
IDoc Services.....	274
IDoc-XML Services.....	280
Monitoring Services.....	283
bXML Services.....	284
BAPI Services.....	286
Transaction Administration Services.....	287
Transport Services.....	292
Specifications.....	302
Sample Services.....	306
SAP Listener Services.....	309
SAP Utility Services.....	311
webMethods Adapter for SAP IDoc Java API.....	311
E Configuration Parameters.....	313
watt.sap.fastTIDCreation.....	314
F Deprecated Services.....	315
List of Deprecated Services.....	316
G Working with Code Pages.....	317
Using Different Code Pages.....	318
H Using BizTalk Envelopes with Adapter for SAP.....	321
Overview.....	322
Use of the BizTalk Header.....	322
Error Handling.....	325
I Using IFR-XML Format with Adapter for SAP.....	329
Overview.....	330
XML Format for BAPIs.....	332
XML Format for RFCs.....	337

XML Format for IDocs.....339

About this Guide

- Document Conventions 10
- Online Information and Support 11
- Data Protection 12

This guide explains how to install, upgrade, and uninstall webMethods Adapter for SAP 10.1, as well as how to configure and develop applications for it. 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 Adapter for SAP and XML.
- Be familiar with the setup and operation of the webMethods Integration Server.
- Have a general idea about how to perform basic tasks with Software AG Designer .
- Know how to create flow services and/or Java services.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format <i>folder.subfolder.service</i> , APIs, Java classes, methods, properties.
<i>Italic</i>	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
Monospace font	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the symbol.
[]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at <https://documentation.softwareag.com>.

In addition, you can also access the cloud product documentation via <https://www.softwareag.cloud>. Navigate to the desired product and then, depending on your solution, go to “Developer Center”, “User Center” or “Documentation”.

Product Training

You can find helpful product training material on our Learning Portal at <https://knowledge.softwareag.com>.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at <https://techcommunity.softwareag.com>. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at <https://github.com/softwareag> and <https://hub.docker.com/u/softwareag> and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at <https://empower.softwareag.com>. Many services on this portal require that you have an account. If you do not yet have one, you can request it at <https://empower.softwareag.com/register>. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Overview of webMethods Adapter for SAP

■ What Is webMethods Adapter for SAP?	14
■ Functional Highlights	15
■ Architecture and Components	18
■ Adapter Package Management	24
■ Adapter Connections	24
■ Adapter Services	26
■ Adapter Listeners and Listener Notifications	26
■ Viewing the Adapter's Update Level	28
■ Controlling Pagination	28

What Is webMethods Adapter for SAP?

webMethods Adapter for SAP allows you to extend your SAP business processes and integrate non-SAP products using open and non-proprietary technology. Adapter for SAP 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 Adapter for SAP**. You can access all SAP functionality that is available via RFC from Adapter for SAP. 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 Adapter for SAP**. You can send Intermediate Documents to Adapter for SAP 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 Adapter for SAP. Adapter for SAP 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**. Adapter for SAP provides rich routing capabilities for BAPIs, RFCs, and IDocs. Different transport types are 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.

Adapter for SAP 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 Adapter for SAP 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 through RFC, tRFC, qRFC, and bgRFC
- Bi-directional and multithreaded communication to and from SAP systems
- Load balancing of incoming SAP documents.
- Support of publishing to the local webMethods Integration Server, to the Broker, or to JMS for asynchronous adapter notifications
- Support of "publish and wait" and "direct service invocation" execution modes for synchronous adapter notifications
- 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

Adapter for SAP 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 Adapter for SAP 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.

Adapter for SAP transparently supports both synchronous (RFC) and asynchronous (tRFC) calls from SAP systems. Thus, BAPIs and ALE scenarios are supported. Adapter for SAP also supports qRFC for IDoc processing and inbound bgRFC calls from SAP.

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 Adapter for SAP 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. Adapter for SAP 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, Adapter for SAP transparently manages communication between different SAP system versions.

Routing IDocs, RFCs, and BAPIs

Adapter for SAP provides rich routing capabilities for synchronous RFCs, asynchronous IDocs, and BAPIs, giving you better control of your trading relationships. Within minutes, you can configure Adapter for SAP 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, and 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

Adapter for SAP 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 Adapter for SAP 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, Adapter for SAP 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 [“Using BizTalk Envelopes with Adapter for SAP” on page 321](#) 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 Adapter for SAP.

For easy identification of the relevant application interfaces, Adapter for SAP 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 Adapter for SAP.

Maintaining Transaction Status Information

Transaction status information can be stored locally or in a centralized repository. Locally stored transaction status information will be accessed only by Adapter for SAP on which the transaction is executed. The transaction status information stored in a central repository will be accessed by the adapters for SAP that are grouped. The available types of storing the transaction status information are:

- Local Transaction Store
- Centralized Transaction Store
- Shared Transaction Store

Local Transaction Store

The Transaction Store is used to store the transaction status information locally. Adapter for SAP 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 webMethods Integration Server. For more information, see [“Managing Transactions and the Transaction Store” on page 178](#).

Centralized Transaction Store

The Centralized Transaction Store (CTS) is a group of adapters for SAP that belong logically together, that share a single Centralized Transaction Store. The CTS is used to store all the processed transactions in a central repository, so that the transaction status information will always be synchronized and valid. The CTS allows reliable tRFC/IDoc load balancing. There can always be just one CTS per adapter group, not two or more. All adapters of SAP that are configured to share a specific CTS are considered to be in the same group. You must configure the Remote Server with alias "SAPGroupStore" for each Adapter for SAP of your adapter group (and in exactly the same way). For more information, see [“Considerations about Adapter for SAP Centralized Transaction Store or Shared Transaction Store” on page 55](#) and [“Centralized Transaction Store” on page 189](#).

Shared Transaction Store

The Shared Transaction Store (STS) is a common transaction repository shared by several adapters for SAP that belong logically together. Similar to CTS, the STS is used to store all the processed transactions in a central repository, so that the transaction status information will always be synchronized and valid. It is a more robust configuration without central server so there is no single point of failure. All adapters for SAP that are configured to share a specific STS are considered to be in the same group. You must configure the location of the STS repository with the configuration switch `watt.sap.xtn.cts.txstore` for each Adapter for SAP of your adapter group (and in exactly the same way). For more information, see [“Considerations about Adapter for SAP Centralized Transaction Store or Shared Transaction Store” on page 55](#) and [“Shared Transaction Store” on page 188](#)

Architecture and Components

Basic Concepts

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

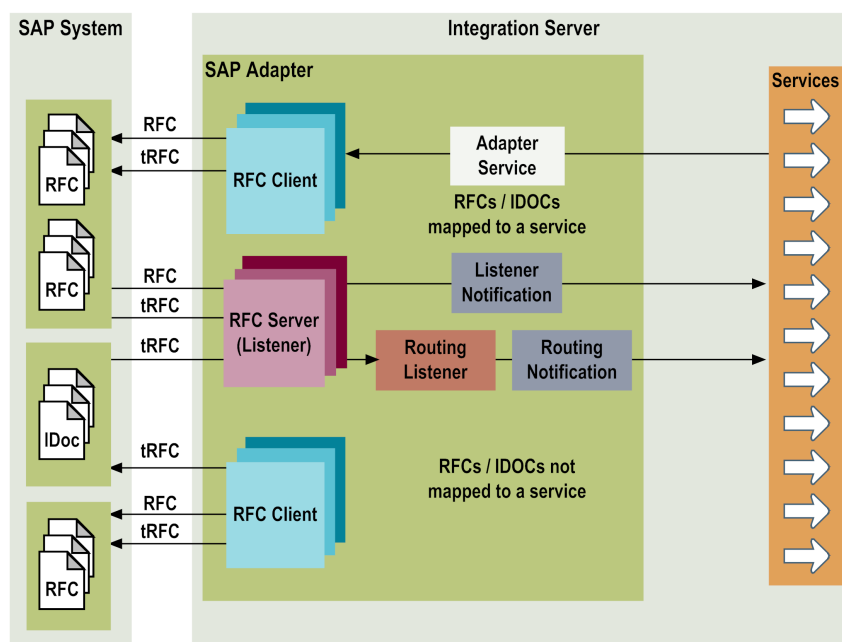
Term	Description
BAPI	Business functions in an SAP system that are written in the programming language ABAP. BAPIs are formalized RFCs. Systems remote from the SAP system commonly use BAPIs to have the SAP system perform an action.
RFC Server (Listener)	<p>SAP terminology for a process that can accept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

Term	Description
	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.
tRFC (Transactional RFC)	Protocol for ensuring that an RFC is successfully executed and executed exactly once on the target system. Adapter for SAP can handle both inbound and outbound tRFCs.
bgRFC (Background RFC)	Protocol for background and queued transactional processing. Adapter for SAP handles inbound bgRFC calls from SAP. Note: To use the support of bgRFC, install webMethods Adapter for SAP Fix 1 and later.
tRFC protocol	Communications method that: <ul style="list-style-type: none"> ■ 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 <p>The transactional RFC (tRFC) protocol ensures that an RFC is successfully executed and that it is executed exactly once.</p>
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 and asynchronously invoke a function on a remote system, and a remote system uses to synchronously and asynchronously 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 provided with webMethods components.
JCo 3 Library	Code library provided by SAP allowing third parties to integrate the RFC protocol into applications. This is how Adapter for SAP 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. Adapter for SAP includes a pre-configured routing listener called <code>wm.sap.internal.ls:routingListener</code> . For more information about the routing listener, see “Overview” on page 141 .
Local Transaction Store	A repository that Adapter for SAP uses to track all transactions that pass through Adapter for SAP.
Centralized Transaction Store	A centralized repository maintained by the CTS server that Adapter for SAP uses to track all transactions that pass through all the adapters of Adapter for SAP group.
Shared Transaction Store	A common shared repository that Adapter for SAP uses to track all transactions that pass through all the adapters of Adapter for SAP group.

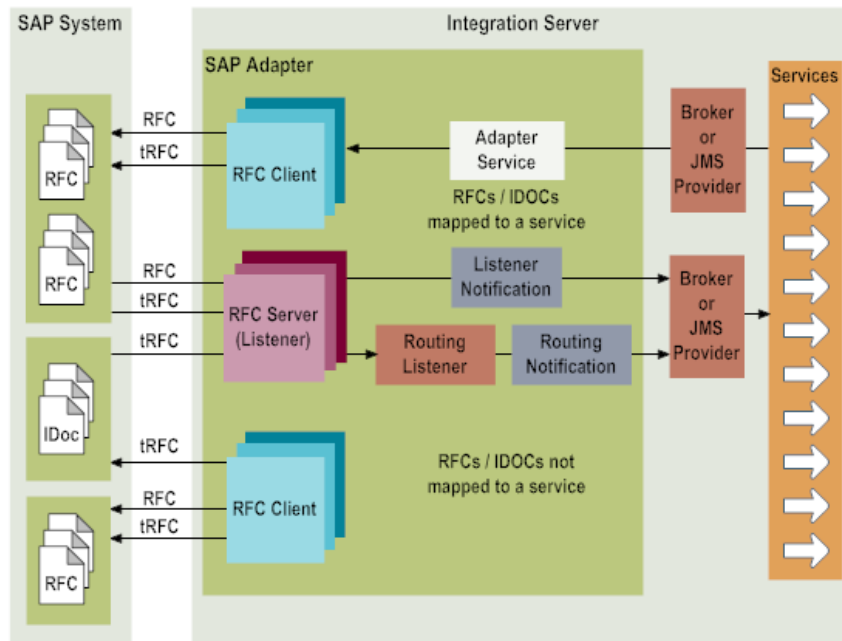
The following illustrates the components in a system that uses Adapter for SAP to communicate between an SAP System and a local Integration Server.

Architecture and Components (Local Integration Server)



The following illustrates the components in a system that uses Adapter for SAP to communicate between an SAP System and a remote Integration Server connected to the local Integration Server by way of either a Broker or a JMS provider:

Architecture and Components (Remote Integration Server)

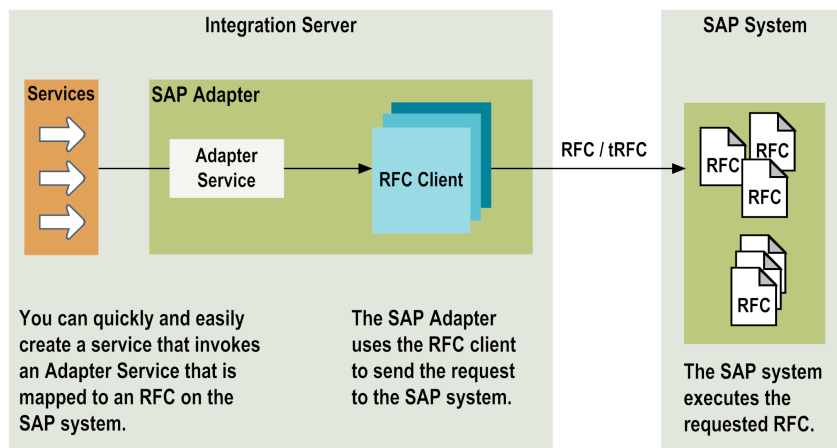


Invoking Business Logic

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

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

Outbound Requests Mapped to an Adapter Service



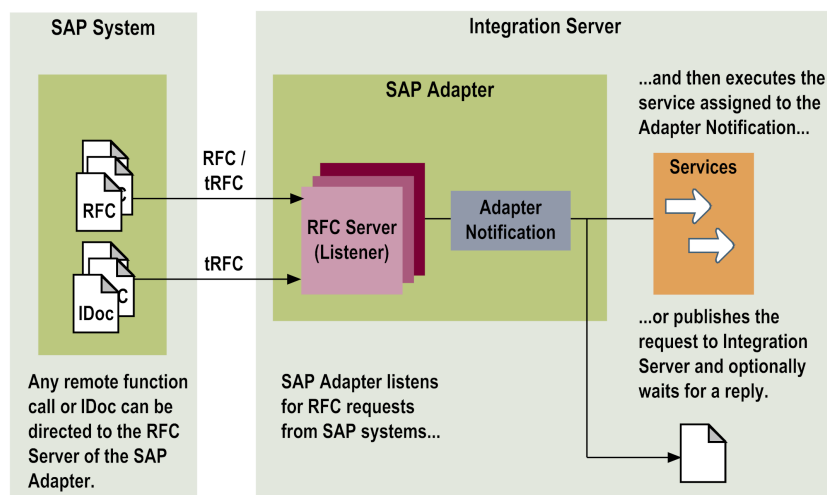
Adapter for SAP uses the RFC Server (Listener) to listen for incoming requests to do one of the following:

- Execute services on Integration Server. Adapter for SAP performs this action for synchronous adapter notifications configured with an execution mode of "service invoke." For more information about execution modes, see ["Service Execution Modes" on page 28](#).
- Publish the request to the local Integration Server (or to a remote Integration Server connected to the host Integration Server by way of a Broker or a JMS provider) and optionally wait for a reply document. Adapter for SAP performs this action for asynchronous adapter notifications and for synchronous adapter notifications configured with an execution mode of "publish and wait." For more information about execution modes, see ["Service Execution Modes" on page 28](#).

From an SAP system point of view, calling Adapter for SAP is no different from calling any other RFC server.

The following diagram illustrates this process when Adapter for SAP is communicating with the host Integration Server. Adapter for SAP can also communicate with a remote Integration Server that is connected to the host Integration Server by way of a Broker or a JMS provider.

Inbound Requests Mapped to an Adapter Notification

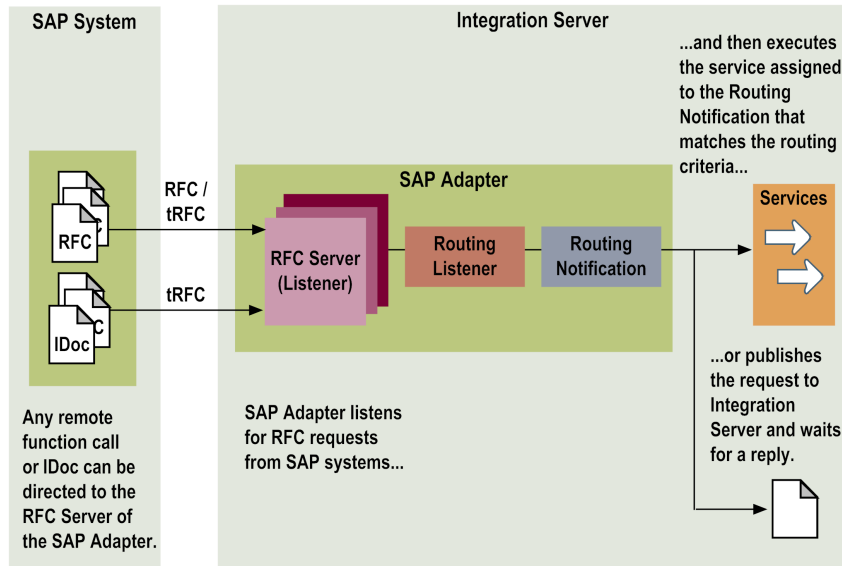


Sending Messages Through the Routing Listener

When Adapter for SAP receives a request on the RFC Server (Listener) that is not associated with a specific adapter notification, Adapter for SAP 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

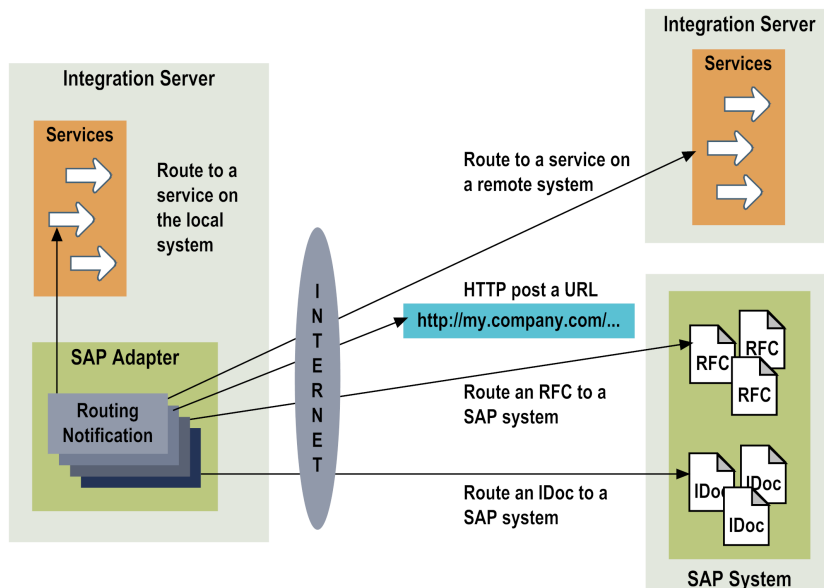


It then uses routing notifications to determine where to route the messages. The routing notification indicates where to route the message based on who the message is from (sender), who is to receive the message (receiver), and the message type.

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

Adapter Package Management

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

Several considerations exist regarding how you set up and effectively manage your packages on the Integration Server, such as the following:

- Configure user-defined packages for your adapter connections and adapter services. See [“Managing the Adapter Package” on page 46](#) 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 47](#) for details.
- Control which development groups have access to which adapter services. See [“Controlling Group Access” on page 50](#) for details.
- Understand how clustering, an advanced feature of webMethods Integration Server, works to effectively manage your adapter services. See [“Using the Adapter in a Clustered Environment” on page 50](#) for details.
- Enable and disable packages. See [“Enabling Packages” on page 47](#) for details.
- Load, reload, and unload packages. See [“Loading, Reloading, and Unloading Packages” on page 48](#).

Adapter Connections

An adapter connection enables Adapter for SAP 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 Adapter for SAP connections, see [“Adapter Connections” on page 57](#). For information about setting user privileges, see *webMethods Integration Server Administrator’s Guide* for your release.

For a list of tasks that you must do before you can create your connections, see [“Before Configuring or Managing Adapter Connections” on page 58](#).

Connection Pools

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. 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, 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 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 [“Adapter Connections” on page 57](#) on [“Adapter Connections” on page 57](#).

Built-In Services for Connections

Integration Server 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 25](#).

Changing the Connection Associated with an Adapter Service at Design Time

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 Designer by selecting the service in one of those tools and running it.

Other built-in services enable you to control connections; for more information, see [“Built-In Services for Connections” on page 25](#).

Changing the Connection Associated with an Adapter Service at Run Time

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 65](#).

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 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 Adapter for SAP. 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 Designer to configure the adapter service. Some familiarity with using these tools is required. For more information, see *webMethods Service Development Help* for your release.

Adapter for SAP 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 Integration Server. It uses the following 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 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 Integration Server Administrator. For detailed instructions on how to configure listeners, see [“Listeners” on page 94](#).

Listener Notifications

A listener notification works in conjunction with a listener to filter and process the RFC requests in Adapter for SAP. Adapter for SAP uses the following 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.
RFC Listener Notification (asynchronous)	Handles incoming RFCs, tRFCs, and bgRFCs asynchronously.
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 “Routing Messages Through Adapter for SAP” on page 139 for more information about routing notifications.

Choice of Publish Destinations

You can choose the destination to which asynchronous notifications should publish messages. Specifically, you can choose whether the asynchronous notification templates use JMS APIs to publish messages to Integration Server or webMethods Broker APIs to publish notification messages to webMethods Broker.

This feature allows you to add asynchronous adapter notification receive steps to a business process model using the same protocol used in existing receive steps in the model.

Note:

To use the JMS protocol with asynchronous notifications, you must first configure a JMS connection alias on Integration Server. For more information, see *webMethods Integration Server Administrator's Guide* for your release.

For steps for selecting a publish destination for asynchronous notification messages, see [“Listener Notifications” on page 103](#).

Service Execution Modes

Synchronous adapter notifications execute flow services and return values from those services. Integration Server supports the following two execution modes to determine how the notification executes a flow service and returns output:

- **Service Invoke mode:** When this mode is selected, the adapter notification invokes a flow service directly and returns values as output from that service.
- **Publish and Wait mode:** When this mode is selected, the adapter notification publishes the received data either to the local Integration Server or to webMethods Broker connected to that Integration Server by converting request documents into publishable documents, and then waits for a reply. This mode is needed when a synchronous adapter notification is represented as a receive step in a process model.

For steps for specifying execution modes for synchronous adapter notifications, see [“Listener Notifications” on page 103](#).

Viewing the Adapter's Update Level

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 Integration Server Administrator.

Controlling Pagination

You can control the number of items that are displayed on the adapter Connections screen, Listeners screen, and Listener Notifications screen. By default, 10 items are displayed per page. Click **Next** and **Previous** to move through the pages, or click a page number to go directly to a page.

To change the number of items displayed per page, set the `watt.art.page.size` property and specify a different number of items.

➤ To set the number of items per page

1. From Integration Server Administrator, click **Settings > Extended**.
2. Click **Edit Extended Settings**. In the Extended Settings editor, add or update the `watt.art.page.size` property to specify the preferred number of items to display per page. For example, to display 50 items per page, specify:

```
watt.art.page.size=50
```

3. Click **Save Changes**. The property appears in the Extended Settings list.

For more information about working with extended configuration settings, see *webMethods Integration Server Administrator's Guide* for your release.

2 System Requirements

■ Overview	30
■ CPU	30
■ Disk Space	30
■ Memory	32
■ Miscellaneous	35

Overview

For detailed information on system requirements, see *webMethods Adapters System Requirements*.

The webMethods Adapter for SAP 10.1 is supported on all platforms where both webMethods Integration Server 10.1 and the SAP Java Connector(SAP JCo) 3.1 are supported. For information on supported platforms for SAP Java Connector, see OSS note 1077727.

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 Adapter for SAP 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 Adapter for SAP 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 *webMethods Integration Server Administrator's Guide* for your release.)

Local Transaction Store

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

(Average size of a transactional message) × (Number of transactions per day) × (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 packages_directory\wmSAP\txStore directory, including its subdirectories. For information about the packages_directory, refer to the section [“The Integration Server Home Directory” on page 38](#) .
2. Send 100 typical transactional messages (transactions you will use in production) through Adapter for SAP.

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 292](#).

Log Files

The log files need additional disk space. If Adapter for SAP 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 Adapter for SAP 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 Integration Server Administrator to turn off audit logging as follows:
 - Go to the **Settings > Logging** page and disable the logger you want to turn off. For more information, see *webMethods Audit Logging Guide* logging guide for your release.
 - If you are using an earlier version of Integration Server, use the **Edit Extended Settings** feature 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 Integration Server and adding the parameter to or editing it in the `Integration Server_directory\instances\instance_name\config\server.cnf` file.

- **Session Log:** Use the Event Manager tool in Designer 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 Adapter for SAP:

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 Integration Server have to handle simultaneously? For example, how many RFCs, HTTP requests, FTP connections, started listeners and RFC listeners will Integration Server have to handle simultaneously at the peak time of the day? This determines the number of sessions n . For more information, see [“Initial Consumption” on page 32](#).
2. For how many SAP systems will Adapter for SAP register an RFC listener? How many threads do these listeners have?
3. How many messages (XML documents, IDocs, RFC calls) go through Integration Server simultaneously at the peak time of the day? Is there any expensive mapping to be done?

Initial Consumption

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” on page 32](#)
- [“Consumption for RFC Listeners” on page 33](#)
- [“Payload” on page 33](#)

Consumption Per Session

Each task Integration Server will handle (incoming HTTP request, incoming/outgoing RFC call) requires one *session* on Integration Server. If n is the number of sessions (see question 1 in the section [“Questions on Sizing” on page 30](#)), add the following amount of RAM:

$\frac{1}{2} n$ MB

If n proves to be much larger than 75, use the **Edit Extended Settings** feature of Integration Server Administrator to view the `watt.server.threadPool` key and edit the key to enable Integration Server to handle more than 75 tasks in parallel. 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 `Integration Server_directory\instances\instance_name\config\server.cnf` file.

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

Note:

This procedure should be used with care to avoid unnecessary memory overload. For example, if you keep a pool of 1000 sessions and 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 in section [“Questions on Sizing” on page 30](#)). 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, 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 in section [“Questions on Sizing” on page 30](#)). 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 = 28KB 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, Integration Server starts with 128MB RAM. After you have calculated the memory that your Adapter for SAP should use, you have to modify the setting:

```
set JAVA_MAX_MEM=128M
```

in *Integration Server_directory\instances\instance_name\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:

- 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.
- 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 $(20\text{KB} \times 6 + 28\text{KB}) \times 3 = 444\text{KB}$.

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 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.

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 too small, you will not take full advantage of the performance improvement. If it is too big, 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 31](#).

3 Installing, Upgrading, and Uninstalling webMethods Adapter for SAP 10.1

■ Overview	38
■ Requirements	38
■ The Integration Server Home Directory	38
■ Installing Adapter for SAP	38
■ Upgrading to Adapter for SAP 10.1	41
■ Uninstalling Adapter for SAP 10.1	43

Overview

This chapter explains how to install, upgrade, and uninstall webMethods Adapter for SAP 10.1. The instructions use Software AG Installer and Software AG Uninstaller wizards. For complete information about the wizards or other installation methods, or to install other webMethods products, see *Installing webMethods Products On Premises* for your release.

Requirements

For a list of the operating systems and SAP products supported by the adapter, see *webMethods Adapters System Requirements*, available in webMethods area of the [Software AG Documentation Web page](#).

In general, Adapter for SAP 10.1 has no hardware requirements beyond those of its host Integration Server. However, see “[System Requirements](#)” on page 29 for a detailed discussion on determining hardware requirements for heavily used adapters for SAP.

The Integration Server Home Directory

With Integration Server 10.1 and higher, you can create and run multiple Integration Server instances under a single installation directory. Each Integration Server instance has a home directory under *Integration Server_directory\instances\instance_name* that contains the packages, configuration files, log files, and updates for the instance.

For more information about running multiple Integration Server instances, see *webMethods Integration Server Administrator's Guide* for your release.

This guide uses the *packages_directory* as the home directory in Integration Server classpaths. The *packages_directory* is *Integration Server_directory\instances\instance_name\packages* directory.

Installing Adapter for SAP

Note:

If you are installing the adapter in a clustered environment, you must install the adapter on each Integration Server in the cluster, and each installation must be identical. For more information about working with the adapter in a clustered environment, see “[Clustering Considerations and Requirements](#)” on page 52.

➤ To install Adapter for SAP 10.1

1. Download Software AG Installer from the [Empower Product Support Web site](#).
2. If you are installing the adapter on an existing Integration Server, shut down Integration Server.
3. Start the Installer wizard.

- Select the webMethods release that includes the Integration Server on which to install the adapter. Adapter for SAP 10.1 requires Integration Server 10.1 or higher and does not run on previous versions.
 - If you are installing on an existing Integration Server, specify the webMethods installation directory that contains the host Integration Server. If you are installing both the host Integration Server and the adapter, specify the installation directory to use. Software AG Installer installs the adapter in the `packages_directory`.
4. In the product selection list, select **Adapters > webMethods Adapter 10.1 for SAP**.
 - If you are want to create RFC adapter services or RFC and IDoc documents, select **Designer > Services > SAP Integration 10.1**. The plug-in for Designer lets you generate webMethods document types based on IDoc and RFC structures defined on an SAP system.

With Integration Server 10.1 and above, you can install the package in the default instance. In this case, Software AG Installer installs the adapter in both locations, *Integration Server_directory\packages* directory and the default instance packages located in *Integration Server_directory\instances\default\packages* directory.

Note:

You can download the adapter documentation at a later time from the [Software AG Documentation Web page](#).

5. After Installer completes the adapter installation, close the Installer.
6. Integration Server requires access to some SAP libraries. Place these libraries in Integration Server classpath as described below.
 - a. Download the latest 3.1 version of the following archive files from the SAP Support Portal of the SAP Service Marketplace:

Archive Files	Download from...
<code>sapjco3-platform-*.*</code>	Connectors > SAP Java Connector > Tools and Services > SAP JCO Release
<code>sapidoc3-*.*</code>	Connectors > SAP Java Connector > SAP Java IDoc Class Library > SAP Java Base IDoc Class Library

Important:

- Use version JCo 3.1.5 (or higher) and IDoc Library 3.1.1 (or higher).
- JCo 3.0.0, IDoc Library 3.0.0, and JCo version 2 are not supported.
- Install the Microsoft security patch as described in SAP note 2786882.
- On Windows platforms, JCo 3.1 requires the Visual Studio 2013 C/C++ runtime libraries to be installed on the system. If not present, download and install the Visual C++ 2013 Redistributable Package from the Microsoft knowledge base article <https://support.microsoft.com/en-us/help/4032938> and select the

package which corresponds to the used Locale and JVM bit-width (x64 for 64-bit or x86 for 32-bit).

- b. Extract libraries from the archive files into the directories specified below:

From zip file	This file	To this directory
sapjco3-platform-*.*	*sapjco3.*	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \lib\
	sapjco3.jar	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmSAP\code\jars\static\
sapidoc3-*.*	sapidoc3.jar	<i>Integration Server_directory</i> \instances\ <i>instance_name</i> \packages\WmSAP\code\jars\static\

- c. For UNIX installations, set the environment variables listed below to JCo library install path:

System	Environment Variable
Linux, Solaris	LD_LIBRARY_PATH
AIX	LIBPATH
HP-UX	SHLIB_PATH

Important:

If older versions of the native libraries are in the path, remove them.

7. Start the host Integration Server.
8. If you are running Integration Server as a service, run register dll command to start Adapter for SAP. For information on installing Microsoft DLLs, see SAP note 0000684106.

JCo 3.1 requires the definition of the services and ports it is using in the etc/services file of the system where it is installed. On Windows, this file is located in the C:\WINDOWS\system32\drivers\etc directory.

The file determines the port and protocol for each service, and it must contain entries for all services used by SAP Java Connector(JCo). For example:

```
sapgw00 3300/tcp
sapgw01 3301/tcp
sapgw02 3302/tcp
sapgw03 3303/tcp
sapgw04 3304/tcp
sapgw05 3305/tcp
...
sapgw00s 4800/tcp
sapgw01s 4801/tcp
sapgw02s 4802/tcp
```



```
sapgw03s 4803/tcp
sapgw04s 4804/tcp
sapgw05s 4805/tcp
```

For more details, read the description in SAP note 0000052959.

Upgrading to Adapter for SAP 10.1

Upgrading from Adapter for SAP 7.1

Adapter for SAP 10.1 is downward compatible with Adapter for SAP 7.1 at the public API level, and it is also fully compatible with the integrations built using Adapter for SAP 7.1 so there is no action required.

Upgrading from Adapter for SAP 6.5

Adapter for SAP 10.1 is compatible with Adapter for SAP 6.5 at the public API level, and it is also compatible with most of the integrations built using Adapter for SAP 6.5.

Please read the following compatibility considerations before upgrading:

- Adapter for SAP 10.1 is compatible with Adapter for SAP 6.5 on the Public Service API; therefore, no action is required for public APIs.
- Adapter for SAP 6.5 uses the previous SAP IDoc Library 2 while Adapter for SAP 10.1 uses the redesigned SAP IDoc Library 3. SAP has changed the package name and internal structure of the SAP IDoc Class Library 3.

The new SAP IDoc Library 3 package name is `com.sap.conn.idoc`, whereas the package name of the old SAP IDoc Library 2 was `com.sap.mw.idoc`. The names of the main IDoc Library classes are also changed as follows:

SAP IDOC Library 3 class...	SAP IDOC Library 2 class
<code>com.sap.conn.idoc.IDocDocumentList</code>	<code>com.sap.mw.idoc.IDoc.DocumentList</code>
<code>com.sap.conn.idoc.IDocDocument</code>	<code>com.sap.mw.idoc.IDoc.Document</code>
<code>com.sap.conn.idoc.IDocSegment</code>	<code>com.sap.mw.idoc.IDoc.Segment</code>
<code>com.sap.conn.idoc.IDocSegmentIterator</code>	<code>com.sap.mw.idoc.IDoc.SegmentIterator</code>
<code>com.sap.conn.idoc.IDocSegmentMetaData</code>	<code>com.sap.mw.idoc.IDoc.SegmentMetaData</code>
<code>com.sap.conn.idoc.IDocRecord</code>	<code>com.sap.mw.idoc.IDoc.Record</code>
<code>com.sap.conn.idoc.IDocRecordMetaData</code>	<code>com.sap.mw.idoc.IDoc.RecordMetaData</code>

The redesign of the SAP IDoc Class Library 3 required the following changes to Adapter for SAP 10.1 IDoc Java API:

Package	Changes
com.wm.adapter.sap.idoc.IDataRecord	Return values and arguments might be of different type due to changes in the IDoc Library package structure. The setField(..) methods have been renamed to setValue(..).
com.wm.adapter.sap.idoc.IDataSegment	Return values and arguments might be of different type due to changes in the IDoc Library package structure.
com.wm.adapter.sap.idoc.IDataDocument	Return values and arguments might be of different type due to changes in the IDoc Library package structure.
com.wm.adapter.sap.idoc.IDataDocumentList	Return values and arguments might be of different type due to changes in the IDoc Library package structure. An IDataDocumentList may contain only IDocs of the same type. The addDocument(..) methods will therefore throw an IDocException if an IDoc of a different type is added to an non-empty IDataDocumentList.
com.wm.adapter.sap.idoc.IDocDocumentList	Return values and arguments might be of different type due to changes in the IDoc Library package structure. IDocDocumentList might contain only IDocs of the same type. Therefore, the addDocument(..) methods have been removed. You must use the add(..) or addNew(..) methods instead.

See the API documentation of the SAP IDoc Library 3 for more details.

Manually Upgrading

The following manual migration instructions address the compatibility considerations listed above.

Public Service API

No action is required for public APIs.

Java IDoc API

Software AG suggests that you replace any usage of the classes from com.wm.pkg.sap.idoc Java package in your custom application code with the corresponding classes from com.wm.adapter.sap.idoc package. For more information about these classes, see webMethods Adapter for SAP IDoc Java API which is available on Adapter for SAP 10.1 home page.

As described in the previous section, the com.wm.adapter.sap.idoc package is changed from Adapter for SAP 6.5 to Adapter for SAP 10.1 as a result of changes in the SAP IDoc Library 3. Therefore, you must modify all Java application code based on the com.wm.adapter.sap.idoc package of Adapter for SAP 6.5 to compile and run properly with Adapter for SAP 10.1. Do the following:

- Replace any references to `com.sap.mw.idoc` with `com.sap.conn.idoc`.
- Replace old SAP IDoc Class Library 2 classes with their IDoc Class Library 3 equivalents described in the table above.
- Rename all `IDataRecord.setField(..)` calls to `IDataRecord.setValue(..)`.
- Place calls to the `IDataDocumentList.addDocument(..)` methods in a try/catch-block to avoid `IDocExceptions` when the IDoc type of the new IDoc document is different from the IDoc type of the other IDocs in the list.
- Substitute `addDocument(..)` methods with calls to the `add(..)` or `addNew(..)` methods because the `addDocument(..)` methods have been removed from the `IDocDocumentList` class.

Uninstalling Adapter for SAP 10.1

➤ To uninstall Adapter for SAP 10.1

1. Shut down the host Integration Server. You do not need to shut down any other webMethods products or applications that are running on your machine.
2. Start Software AG Uninstaller, selecting the webMethods installation directory that contains the host Integration Server. In the product selection list, select **Adapters > webMethods Adapter 10.1 for SAP**.
3. After Uninstaller completes, restart the host Integration Server.
4. Uninstaller removes all Adapter for SAP 10.1 related files that were installed. However, it does not delete the files created after you installed the adapter, nor does it delete the adapter directory structure. You can go to the `packages_directory` and delete the `WmSAP` directory. On Integration Server 10.1 and above, go to `Integration Server_directory\instances\instance_name\packages` directory and delete the `WmSAP` directory.
5. Uninstaller does not delete any user-defined Adapter for SAP 10.1 components such as connections, adapter services, or adapter notifications. Because these components will not work without the adapter, delete them manually, either at the file system level or using Designer. For instructions, see *webMethods Service Development Help* for your release.

4 Package Management

■ Overview	46
■ Managing the Adapter Package	46
■ Controlling Group Access	50
■ Using the Adapter in a Clustered Environment	50

Overview

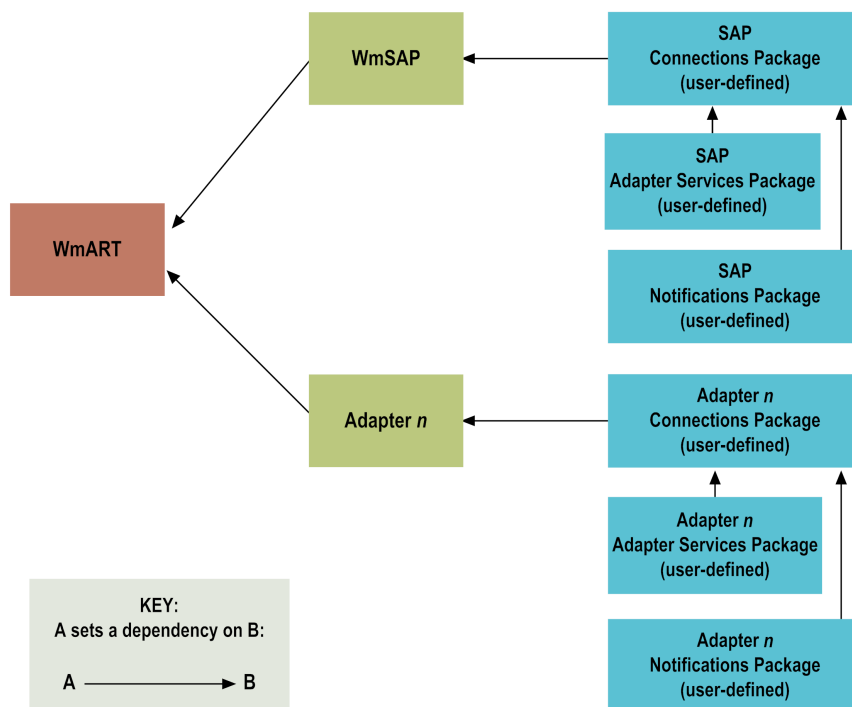
The following sections describe how to set up and manage your Adapter for SAP packages, set up Access Control Lists (ACL), and use the adapter in a clustered environment.

Managing the Adapter Package

Adapter for SAP is provided as a package called WmSAP. You manage the WmSAP package as you would manage any package on 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 Designer 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 47)
- [“Enabling Packages”](#) on page 47
- [“Disabling Packages”](#) on page 48
- [“Importing and Exporting Packages”](#) on page 49

- [“Controlling Group Access” on page 50](#)

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 *webMethods Service Development Help* for your release.

- 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 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 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.
- 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 Packages” on page 47](#).
- 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 [“Disabling Packages” on page 48](#).
- You can name adapter services and notifications the same name provided that they are in different folders.

Enabling Packages

All packages are automatically enabled by default. However, if a package has been previously disabled, you can enable it following the procedure below.

➤ To enable a package

1. Open 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 48](#).

Important:

Before you manually enable a user-defined package, you must first enable its associated adapter package (WmSAP).

Disabling Packages

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 Integration Server from loading that package at startup.

➤ To disable a package

1. Open 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 disable 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 Integration Server Administrator.
- Not be listed in the Software AG Designer Package Navigator view.

Important:

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.

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 47](#)), 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 Integration Server Administrator (by clicking the **Reload** icon on the Management window) or from Software AG Designer (by right-clicking the package and selecting the **Reload Package** option from the menu).

Unloading Packages

At shutdown, 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 Designer. 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 *webMethods Service Development Help* for your release.

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 Adapter for SAP services should identify webMethods Adapter for SAP 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 *webMethods Service Development Help* for your release.

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 *webMethods Service Development Help* for your release.

Using the Adapter in a Clustered Environment

What Is webMethods Integration Server Clustering?

Clustering is an advanced feature that substantially extends the reliability, availability, and scalability of 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:

- **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 *webMethods Integration Server Clustering Guide* for your release.

Important:

If you use pure Integration Server clustering with Adapter for SAP, the load balancing and failover mechanism of the IS cluster do not apply for Adapter for SAP RFC connections and RFC listeners. The RFC protocol has some restrictions that affect failover and load balancing:

- A (logical) RFC connection is bound to the low-level TCP/IP connection on which it was initiated, so you cannot continue and redirect an RFC connection to a different Integration Server in case the IS that started the RFC connection suddenly dies. Therefore a reliable failover is not supported for RFC connections.
- SAP Transactions that spawn over several transaction steps must be executed over the same (physical) RFC connection. Therefore reliable failover and continuation of a failed multi-step SAP transaction on a different Integration Server is not possible.
- Load balancing of inbound RFC connections is achieved by the SAP Gateway which is independent of Integration Server.

For these reasons, an Integration Server cluster does not provide benefits for SAP integration in comparison to a group of independent IS servers.

About Clustering for Adapter for SAP

The evaluation of clustering has shown that its advantages in the specific SAP environment are restricted. As a result, it is generally recommended 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 light of these restrictions, clustering with an external load balancer offers the following advantages and disadvantages.

Advantages:

- 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 is reached by adding an external load balancer in front of independent Integration Servers to offer better system performance.

Disadvantages:

- If the cluster requires load balancing, an external load balancer is required.
- 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 be 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 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 more information about this service, see [“pub.sap.client:lockSession” on page 260](#).
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 261](#).

How Inbound RFCs Are Balanced

SAP listeners in Adapter for SAP 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 Adapter for SAP to create adapter services, you must ensure that each Integration Server in the cluster contains an identical set of packages.

Replicating Packages to webMethods Integration Servers

Every webMethods Integration Server in the cluster should contain an identical set of packages that you define using Adapter for SAP; that is, you should replicate Adapter for SAP 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 *webMethods Integration Server Administrator's Guide* for your release.

Clustering Considerations and Requirements

Note:

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

The following considerations and requirements apply to Adapter for SAP 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 For example... in a given cluster must have identical...

Integration Server versions	All Integration Servers within a cluster must have exactly the same version. Also, the same service pack and fixes (updates) must have been 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 the adapters for SAP must be the same version, with the same fixes (updates), the same SAP libraries (versions) applied, and the various "watt.sap.*" configuration switches should be identical.
User-defined adapter packages	<p>Each package that you have configured containing adapter connections, adapter services, flow services, listeners, triggers, and notifications for Adapter for SAP, must exist on all Integration Servers in the cluster, so that all Integration Servers in the cluster can handle any given request.</p> <p>The configuration and settings of all Adapter for SAP connections, adapter services, listeners, triggers and notifications must be identical throughout the cluster.</p> <p>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 <i>webMethods Integration Server Administrator's Guide</i> for your release.</p>

Note:

If you plan to use connection pools in a clustered environment, see ["Considerations When Configuring Connections with Connection Pooling Enabled" on page 54](#).

**All Integration Servers For example...
in a given cluster must
have identical...**

Adapter connections	<p>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.</p> <p>If you plan to use connection pools in a clustered environment, see “Considerations When Configuring Connections with Connection Pooling Enabled” on page 54.</p>
Adapter services	<p>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.</p> <p>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.</p>

See [“Replicating Packages to webMethods Integration Servers” on page 52](#) for information about replicating adapter packages, connections, and adapter services across multiple Integration Servers in a cluster.

Considerations When Installing Adapter for SAP Packages

For each Integration Server in the cluster, use the standard Adapter for SAP installation procedures for each machine, as described in [“Installing, Upgrading, and Uninstalling webMethods Adapter for SAP 10.1” on page 37.](#)

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 Adapter for SAP, see [“Configuring Adapter Connections” on page 58](#).

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

Considerations about Adapter for SAP Centralized Transaction Store or Shared Transaction Store

When using multiple instances of Adapter for SAP for inbound RFC load balancing, you configure RFC listeners with the same Program ID on each Adapter for SAP so that the SAP Gateway can perform load balancing and route tRFC documents to different adapters for SAP. The SAP Gateway also sends out tRFC transaction status information updates to Adapter for SAP listeners. By default, this transaction status information is persisted locally in the adapter's transaction store.

However, communication failures, for example when a network connection fails or an Integration Server becomes unavailable, can lead to situations where the SAP Gateway is sending the tRFC transaction status information not to the single Adapter for SAP that has processed the tRFC, but also to other, different adapters for SAP in the group. In this case, the transaction status information persisted in the local transaction stores of the various adapters can end up in an inconsistent and invalid state.

To prevent these synchronization issues with tRFC transaction information updates, you can configure your group of adapters for SAP to use a single Shared Transaction Store (STS) or Centralized Transaction Store (CTS) that hosts all transaction state information for all adapters for SAP in the group.

The CTS implements a client/server solution, where one Adapter for SAP hosts the centralized store (the CTS Server) and all other adapters for SAP access this store as clients (CTS clients). The Shared Transaction Store implements a direct shared store. Since all the transaction status information updates are now routed to a single transaction store, no more duplicate, incomplete, or inconsistent transaction state information will be persisted in the local transaction stores.

You do not have to set up an STS or CTS as an Integration Server cluster. Instead, you can use the STS either with an Integration Server cluster or with a group of adapters for SAP on independent Integration Servers.

An STS (or CTS) is needed when you want to implement robust inbound or outbound load balancing using tRFCs and IDocs. If you just run a single Adapter for SAP, then an STS (or CTS) will not make any sense. If you run a group of completely independent adapters for SAP and do not intend to use load balancing or tRFCs or IDocs with these adapters, then configuring a CTS will not be useful.

Adapters for SAP configured as CTS Clients will persist and look up all the transaction information in the CTS, therefore the availability and performance of the CTS Server is of importance. Ensure that the Adapter for SAP that hosts the CTS (the CTS Server) is always started before all other adapters for SAP (clients). Also, ensure that the CTS server is not shut down or disabled while other CTS clients are still running and processing transactions.

The STS server will be a single point of failure. WmSAP 10.1 therefore introduces the more robust and more performant STS as a substitute for CTS. The CTS configuration is deprecated and should be changed to an STS configuration.

For more information about configuring the Shared and Centralized Transaction Store, see [“Shared Transaction Store” on page 188](#) and .

5 Adapter Connections

■ Overview	58
■ Before Configuring or Managing Adapter Connections	58
■ Configuring Adapter Connections	58
■ Setting Up the SAP System for SNC Connections	64
■ Dynamically Changing a Service's Connection at Run Time	65
■ Enabling Adapter Connections	66
■ Viewing Adapter Connections	66
■ Editing Adapter Connections	68
■ Copying Adapter Connections	68
■ Deleting Adapter Connections	69
■ Disabling Adapter Connections	69
■ Testing the Execution of an RFC	70
■ Testing the Execution of a BAPI Via XML	71

Overview

This chapter describes how to configure and manage Adapter for SAP connections. For more information about how adapter connections work, see [“Adapter Connections” on page 24](#).

Before Configuring or Managing Adapter Connections

➤ To prepare to configure or manage adapter connections

1. Install webMethods Integration Server and Adapter for SAP on the same machine. See [“Installing, Upgrading, and Uninstalling webMethods Adapter for SAP 10.1” on page 37](#) for details.
2. Make sure you have webMethods administrator privileges so that you can access Adapter for SAP's administrative screens. For information about setting user privileges, see *webMethods Integration Server Administrator's Guide* for your release.
3. Start your Integration Server and Integration Server Administrator, if they are not already running.
4. Using Integration Server Administrator, make sure the WmSAP package is enabled. See [“Enabling Packages” on page 47](#) for instructions.
5. Using Designer, 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 46](#).

Configuring Adapter Connections

Use the following procedure to define the parameters that Adapter for SAP will use to establish a connection to an SAP system. Adapter for SAP requires a connection to the SAP system whenever functionality from the SAP system is to be invoked; that is whenever Adapter for SAP acts as a client for an SAP system. But also when Adapter for SAP 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 **Adapter for SAP**.
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 **Adapter for SAP** section:

Field	Description/Action
Package	<p>Package in which to create the connection. You must create the package using Designer before you use in this parameter. For information about creating packages, see <i>webMethods Service Development Help</i> for your release.</p> <p>Note: Configure the connection in a user-defined package rather than in the adapter's package. See “Package Management” on page 45 for other important considerations when creating packages for Adapter for SAP.</p>
Folder Name	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	<p>RFC connection name. This is the name by which the SAP system are known to Adapter for SAP developers, clients, and partners.</p> <p>Note: The Connection Alias must be unique for each connection.</p>
User Name	<p>SAP default user for this RFC adapter connection. If no other SAP user name is provided during runtime then this SAP user account is used to execute the client request in the SAP System.</p> <p>Note: The default user of Adapter for SAP Connection is also used to execute lookups to the DDIC for RFC, BOR and ALE metadata. So, it is necessary to set an SAP logon user with the authorization to the following SAP standard function groups: RFC1, SDIF, SG00, and SRFC.</p> <p>It is a recommended best practice to set a technical SAP user account with the minimum necessary set of authorizations as the SAP default user. If the execution of an RFC adapter service requires additional access rights then the name of an SAP user with sufficient rights must be passed during runtime.</p>
Password	SAP password of the default user.
Repository User Name	SAP repository user for this RFC adapter connection. If a repository user is provided, all meta data and repository lookups are executed with this user account. This field is optional.

Field	Description/Action
Repository Password	Password of the SAP user specified in the Repository User Name field.
Client	Three-digit SAP client number.
Language (optional)	SAP language code. This field is optional. 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:

- Complete the following fields to set up the group logon:

Field	Description/Action
Load Balancing	<ul style="list-style-type: none"> ■ Select On to activate the group login concept. Complete the remaining fields in the section. ■ Select Off to connect to one dedicated application server. Skip to "Complete the following fields to set up the single server logon:" in step 6.
Logon Group	Name of the group you want to login.
Message Server Host	Host name or IP address of the message server.
Message Server Service	Optional. Message server service or port number. Message Server Service can be used instead of the System ID field.
System ID	Optional. System ID of the SAP system or SID. System ID field can be used instead of the Message Server Service field.

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

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

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

Important:

- The settings must point either to the Gateway at which this RFC server is registered or to the **Repository Server** selected.
- The load balancing and single server logon settings are ignored if you configure an RFC server.

Field	Description/Action
External RFC Server	<ul style="list-style-type: none"> ■ Select On to use an external RFC server. Complete the remaining fields in the section. ■ Select Off if you do not want to connect use an external RFC server.
Program ID	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.
Gateway Service	<p>Value corresponds to the system number of the RFC Server. The format is sapgwXXX, where XX is a value from 00 to 99.</p> <p>For example: sapgw07 or sapgw00s</p>
Repository Server	Value is the SAP server 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 Adapter for SAP 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 Adapter for SAP, 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. Complete the following fields to enable security(Optional):

Field	Description/Action
SNC Enabled	<p>Determines whether this server should use Secure Network Communications (SNC). Possible Values are:</p> <ul style="list-style-type: none"> ■ Yes. ■ No. Default
SNC Quality of Service	<p>SNC Quality of service, possible values:</p> <ul style="list-style-type: none"> ■ Use global built-in default settings.

Field	Description/Action
	<ul style="list-style-type: none"> ■ 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 Personal Security Environment (PSE).
SNC Partner Name	SNC name of the SNC partner (RFC server) or SNC name of the message server (Load Balancing).
SNC Authentication	<p>Select one of the following possible SNC authentication types:</p> <ul style="list-style-type: none"> ■ User and Password. Uses the username and password fields for authentication. If the SNC mode is enabled, then SNC is specifically used to encrypt the connection data and not for authentication. SSO authentication is not used. ■ SNC Credential. Uses the credential in the SNC PSE file for the authentication if the SNC mode is enabled. <p>Note: At runtime, if a valid X.509 certificate is available in the current HTTPS Integration Server session, the X.509 certificate takes precedence over the credentials in the SNC PSE file.</p> <ul style="list-style-type: none"> ■ X.509 Certificate. Uses the X.509 certificate available in the current HTTPS Integration Server session for authentication. <p>Note: At runtime, if a valid X.509 certificate is not available in the current HTTPS Integration Server session and the SNC mode is enabled, the credentials in the SNC PSE file is used for authentication.</p>

8. Optionally, set SAP Router String:

Field	Description/Action
SAP Router String	<p>SAP router string. The 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.</p> <p>For example: /H/127.0.0.1/H/, where H indicates the host name.</p>

Field	Description/Action
	The SAP router string is only needed if a firewall exists between the SAP System and Integration Server. For more information on configuring the SAP router, go to the SAP Service Marketplace at: http://service.sap.com/connectors .

9. Optionally, complete the advanced settings for the connection:

Field	Description/Action
Use SAPGui	<p>Run with/without/hide the SAP GUI between two RFC functions. Possible values are:</p> <ul style="list-style-type: none"> ■ Off. Default. Run without SAP GUI between two RFC functions. ■ On. Run with SAP GUI between two RFC functions. ■ Hidden. Run an invisible SAP GUI between two RFC functions.
RFC Trace	<p>Enables the creation of RFC trace information for this client connection. Possible values are:</p> <ul style="list-style-type: none"> ■ Off. Default. ■ On. <p>For detailed information on using the RFC trace, see Logging and Monitoring.</p>
Log Transaction Status	<p>Switch to save the processed logs. Possible values are:</p> <ul style="list-style-type: none"> ■ Off. Default. Processed logs are not saved, although a transaction is created (or maintained), and the transaction can be monitored later on in the transaction list. <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>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 Adapter for SAP.</p> </div> <ul style="list-style-type: none"> ■ On.
Store Message Body	<p>Switch to store the message body of the incoming document to disk. Possible values are:</p> <ul style="list-style-type: none"> ■ Off. Default. Message body of the incoming document is not stored to disk, although a transaction is created (or maintained), and the transaction can be monitored later on in the transaction list. <div style="background-color: #f0f0f0; padding: 10px; margin: 10px 0;"> <p>Tip:</p> </div>

Field	Description/Action
	Setting this switch to Off 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 Adapter for SAP.
	■ On.

Note: Adapter for SAP 10.1 does not support the ABAP Debugger utility. Instead, use the external debugger provided with SAP 6.2 and later versions.

10. Click **Save Connection** to commit these changes.

Setting Up the SAP System for SNC Connections

If you want to use SNC connections with the SAP system, you must configure the SAP system to accept both SNC and non-SNC connections from Adapter for SAP.

When connecting to the SAP system, Adapter for SAP creates two types of connection pools for each connection alias: an SAP repository connection pool and an SAP runtime connection pool. The following table describes each adapter connection pool.

Connection Pool Type	Description
Repository	<ul style="list-style-type: none"> ■ Used internally for metadata lookup ■ No business data is processed over the connections ■ Supports both RFC and SNC connections
Runtime	<ul style="list-style-type: none"> ■ Managed by the Adapter Runtime ■ All business data is processed over the connections ■ Supports both RFC and SNC connections

To use SNC connections, you must configure the following settings in the SAP system.

SAP system settings for SNC	Table/Parameter
General SAP system settings for SNC:	Table SNCSYSACL (SM30, view VSNCSYSACL, TYP=E) SNC name entry of Adapter for SAP.
Certificate log in: General Settings:	<ul style="list-style-type: none"> ■ Profile Parameter: ■ SNC/libsapsecu=../libsecude.sl (for HP-UX)

SAP system settings for SNC	Table/Parameter
	<ul style="list-style-type: none"> ■ SNC/extid_login_diag=1 ■ SNC/extid_login_rfc=1 ■ Table SNCSYSACL (SM30, view VSNCSYSACL, TYP=E), activate: <ul style="list-style-type: none"> ■ Certificate log in ■ Diag ■ RFC
Certificate log in: User specific settings:	<p>Table USREXTID (view VUSREXTID), Category DN</p> <p>Entry with DN (Distinguished Name) from user certificate</p> <p>The Seq.No. 000 is the default entry.</p> <p>Set entry active.</p>

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

Dynamically Changing a Service's Connection at Run Time

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

The adapter services that ship with Adapter for SAP in the WmSAP package in the `pub.sap` namespace include a field called `serverName`. By using this field, you can specify at run time the SAP system to be updated.

Additionally, Integration Server 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 metadata, RFCs, and BAPIs.

For more information, see [“Changing the Connection Associated with an Adapter Service at Run Time” on page 26](#).

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 Integration Server 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 Integration Server Administrator's navigation area, click **Adapter for SAP**.
2. On the Connections screen, click **No** in the **Enabled** column for the connection you want to enable.

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 68](#).
- In case of an RFC error message, an RFC trace file named `dev_rfc.trc` will be written to *Integration Server_directory* directory. This file contains a more detailed description of the error cause.

Viewing Adapter Connections

You can view adapter connections and each connection's parameters from the Integration Server Administrator or from Designer.

Viewing Adapter Connections Using Integration Server Administrator


➤ To view the adapter connections using Integration Server Administrator

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

You can sort and filter the list of connections that appears on the Connections screen.

- To sort information on the Connections screen, click the **Up** and **Down** arrows.
- To filter the list of connections:
 1. On the Connections screen, click **Filter Connections**.
 2. Type the criterion by which you want to filter into the **Filter criteria** box. Filtering is based on the node name, not the connection alias. To locate all connections containing specific alphanumeric characters, use asterisks (*) as wildcards. For example, if you want to display all connections containing the string "abc", type *abc* in the **Filter criteria** box.
 3. Click **Submit**. The Connections screen displays the connections that match the filter criteria.
 4. To re-display all connections, click **Show All Connections**.

The Connections screen appears, listing all the current connections. You can control the number of connections that are displayed on this screen. For more information, see [“Controlling Pagination” on page 28](#).

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

The View Connection screen displays the parameters for the connection. For descriptions of the connection parameters, see [“Configuring Adapter Connections” on page 58](#).

3. Click **Return to Adapter for SAP Connections** to return to the Connections screen.

Viewing Adapter Connections Using Designer

➤ To view the adapter connections using Designer

1. Start Designer if it is not already running.
2. From Designer Package Navigator view, open the package and folder in which the connection is located.
3. Double-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 58](#).

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 Integration Server 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 Integration Server Administrator's navigation area, click **Adapter for SAP**.
2. Make sure that the connection is disabled before editing it. See [“Disabling Adapter Connections” on page 69](#) for instructions.
3. On the Connections screen, click the  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 58](#).

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

Copying Adapter Connections

You can copy an existing Adapter for SAP 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 Integration Server Administrator.

» To copy a connection

1. In the **Adapters** menu in the Integration Server Administrator's navigation area, click **Adapter for SAP**.
2. On the Connections screen, click  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 58](#).


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 Adapter for SAP connection, you can delete it by following the instructions in this section. You delete adapter connections using Integration Server 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 25](#).

➤ To delete a connection

1. In the **Adapters** menu in Integration Server Administrator's navigation area, click **Adapter for SAP**.
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 the  icon for the connection you want to delete.

The Integration Server deletes the adapter connection.

Disabling Adapter Connections

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

➤ To disable a connection

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

➤ To test an RFC from Adapter for SAP

1. In the **Adapters** menu in the Integration Server Administrator navigation area, click **Adapter for SAP**.
2. In **Adapter for SAP** 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**. Adapter for SAP 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**. Adapter for SAP 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 of this procedure.
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 Adapter for SAP 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 Adapter for SAP

1. In the **Adapters** menu in the Integration Server Administrator navigation area, click **Adapter for SAP**.
2. In **Adapter for SAP** 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.
Function module	The Function module 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.
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	<p>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.</p> <p>Note: Key fields are used with instance-dependent (non-static) methods and with factory methods.</p> <p>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.</p> <p>For example, using the key field <code>CompanyCodeId</code> in a call to the <code>CompanyCode.GetDetail</code>:</p>

Field	Description
	<pre> <?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> </pre>

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 [“Routing BAPIs Through Adapter for SAP ” on page 160](#).

6 Using Command Central to Manage Adapter for SAP

■ Configuration Types	78
■ Working with Adapter for SAP Configuration Types	78

Configuration Types

The following is the configuration type for Adapter for SAP:

Configuration Type	Configuration ID	Use to configure...
webMethods Adapter for SAP	ADAPTER-CONNECTIONS	The connection for Adapter for SAP.
	ADAPTER-LISTENERS-NOTIFICATION	The state of the listener notification. You can only update the state of the notification.
	ADAPTER-LISTENERS	The listeners for Adapter for SAP
	ADAPTER-ROUTING-MAPPING	Routing and Mapping for documents.
	ADAPTER-SAP-USERS	To manage the SAP User Store.
	ADAPTER-SETTINGS	The general settings of Adapter for SAP

Working with Adapter for SAP Configuration Types

Command Central supports the Integration Server version 10.4 and later, to edit the configuration for Adapter for SAP 10.1.


Perform the following procedure to add, edit, view, or delete items for Adapter for SAP configuration type items over Command Central.


➤ To create, edit, view, or delete an item for an Adapter for SAP configuration type

1. Select the Integration Server environment from the Environment pane, then click the **webMethods Adapter for SAP** from the **Instances** tab.
2. Click **Configuration** tab
3. Command Central displays the following screens from the drop- down for Adapter for SAP configuration type:

- **ADAPTER-CONNECTIONS**
- **ADAPTER-LISTENERS**
- **ADAPTER-LISTENER-NOTIFICATIONS**
- **ADAPTER-ROUTING-MAPPING**
- **ADAPTER-SAP-USERS**

■ ADAPTER-SETTINGS

4. To create a Connection, Listener, Routing/Mapping, and SAP User for Adapter for SAP configuration type, click . Enter the required values in the displayed fields and click **Save**.

For ADAPTER-ROUTING-MAPPING configuration, click  to display the list of the Routing/Mapping types available. The types are as follows:

Routing/Mapping type	Auto-populated fields	The fields you configure...
ALE Routing Info Default	Service type	Service name
	For Document type	
	Message type	
ALE Routing Info	Service type	Message type
	For Document type	Service name
ALE Mapping Info Default	Service type	Service name
	For Document type	
	Message type	
ALE Mapping Info	Service type	Message type
	For Document type	Service name
XML Routing Info	Service type	Service name
	For Document type	
	Message type	


Note:

By default, the **Active** field contains the Yes radio button enabled.

Note:

For more information about the usage and field descriptions of the Adapter for SAP configuration types, see the following:

- [“Adapter Connections” on page 57](#)
- [“Listener Notifications” on page 103](#)
- [“Listeners” on page 94](#)
- [“Routing Messages Through Adapter for SAP” on page 139](#)
- [“Managing SAP User Store” on page 225](#)
- [“General Adapter for SAP Settings” on page 250](#)

5. To edit any configuration, click the corresponding configuration that you want to update and click **Edit**. Make the necessary changes and click one of the following:
 - **Test** to validate only the field level values.
 - **Save** to save your changes.
 - **Cancel** to cancel the edits to the configuration.
 - To enable the connection, select the Yes radio button in the **Enabled** field of the **Connection State** section. By default, No radio button is selected.
 - Before you edit the fields in **ADAPTER-LISTENERS**, the listener have to be in the **Disabled** state
 - In the **Notification Order Details** section, the **Edit Notification Order** is disabled, by default. To enable it, click on Yes radio button.
 - The **Notification Order** field can be re-ordered manually. But, you cannot modify the existing listening notifications.
 - If you do not have any listener notification, then you cannot edit the field.
 - .
6. To view configuration details, click the respective configuration.
7. To delete a configuration, select the configuration that you want to delete and click .

7 Adapter Services

■ Overview	82
■ Creating an Adapter Service that Executes an RFC	82

Overview

Adapter services for Adapter for SAP 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 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:

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

Terminology

To use Adapter for SAP 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 <i>function interface</i> .

Term	Description
Import	Input parameter to an RFC. Input parameters are simple values (such as string or number) or structures.
RFC Adapter Service	Adapter for SAP terminology for an association between a function module on an SAP system and an adapter service based on the RFC Adapter Service template on Adapter for SAP.
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: Adapter for SAP outbound call is an inbound call from the SAP system's point of view.

➤ To create an adapter service for an RFC

1. Start Software AG Designer.
2. Select **New >Adapter Service**, and do one of the following:
 - If you are using Designer, be sure the parent namespace is selected and type a name for the adapter service. Click **Next**.
3. Select **Adapter for SAP** 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)** and do one of the following:
 - If you are using Designer, click **Finish**.
6. 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 Adapter for SAP 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.

7. Click **Save** 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 Designer , 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 Designer.

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

Important:

If the execution of the RFC adapter service requires additional access rights or if the adapter service should be executed with a specific SAP user account, you can override the SAP default user and provide the user name and password of a different SAP user account as parameter `$user` and `$pass` in the pipeline. These parameters must be set in the pipeline before the adapter service is invoked.

Important:

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. 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, unselect **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. (For more information, see *webMethods Integration Server Administrator's Guide* for your release.)

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

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

Invoking the RFC Adapter Service from a Browser

> 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: ssid=<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 Adapter for SAP 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`.

Invoking the RFC Adapter Service Using the `pub.client:http` Service

➤ To use the `pub.client:http` service

1. Using Designer , navigate to the `pub.client:http` service.
2. Select **Test > Run**.
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 *webMethods Integration Server Built-In Services Reference* for your release.

8 Adapter Notifications

■ Overview	88
■ Creating an RFC Destination on an SAP System	91
■ Listeners	94
■ Listener Notifications	103
■ Examples	120

Overview

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

After you have the SAP system and Adapter for SAP 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 Adapter for SAP for further synchronous or asynchronous processing. To do so, you also need to create a listener notification on Adapter for SAP. The listener notification indicates either what service the adapter is to execute when it receives a message from the SAP system or how to publish a corresponding document.

Adapter for SAP supports the following types of notifications that can be assigned to an RFC listener.

- RFC listener notification (synchronous)
- RFC listener notification (asynchronous)
- 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 [“Routing Messages Through Adapter for SAP” on page 139](#).

Note:

In this context, synchronous or asynchronous refers to the processing on 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 Adapter for SAP”](#) on page 183.

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 Adapter for SAP 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 Adapter for SAP should determine how to proceed after successfully executing and committing the transactional request.

If the Forward Confirm event flag is activated, Adapter for SAP 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 Servers 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, Adapter for SAP 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 of this check table in scenarios like:

- SAP system A->Adapter for SAP A ->Internet->Adapter for SAP B->SAP system B or
- external client->Adapter for SAP->SAP system

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

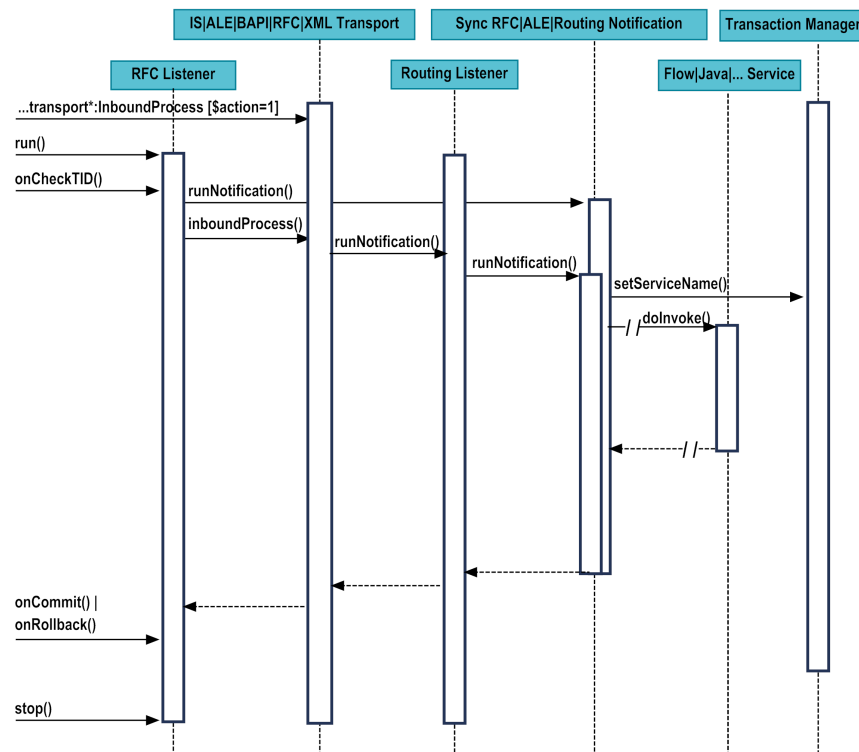
The following two simplified dynamic model diagrams show how Adapter for SAP handles requests differently depending on the setting of the `$action` parameter. They show a combined view of three possible scenarios:

- 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.

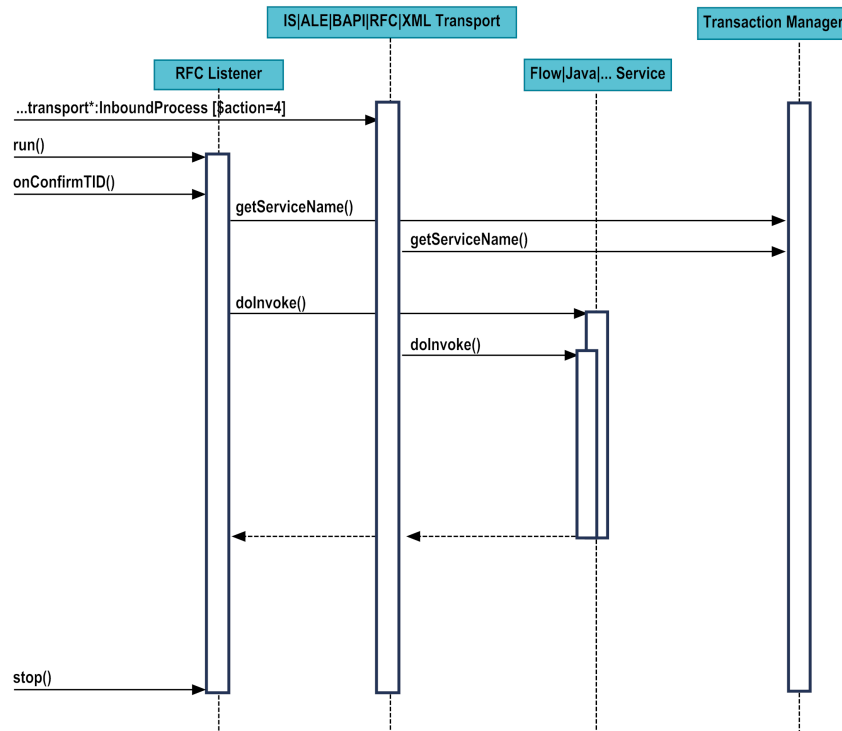
Action Equals 1

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 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).

**Important:**

If you execute a routing notification that has a transport other than 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 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 Adapter for SAP that identifies where the SAP system sends all RFCs that invoke services on Integration Server.

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

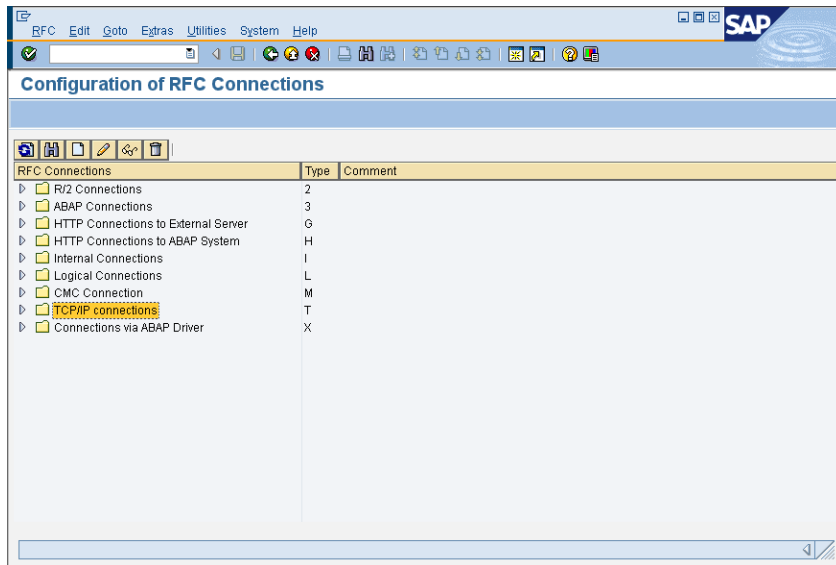
Note:

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 Adapter for SAP listener as an RFC destination

1. Use the SAPGui to log on 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 Adapter for SAP and the SAP system itself. For example, if the SAP system is named CER and 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 recommended that you pick a name that contains 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 Adapter for SAP 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_systemapplicationserver` in the **Gateway host** field.

14. Enter `sapgwnn` (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. If you want the RFC server to run in Unicode mode then select the **Unicode** option on the **Special Options** tab.
16. If you want to use the RFC destination for bgRFC calls then select **basXML** as the **Transfer Protocol** on the **Special Options** tab.

Note:

The Classic with bgRFC and Classic with tRFC transfer protocols are not supported for bgRFC by Adapter for SAP.

An RFC Destination setup for bgRFC should not be used for RFC or tRFC calls.

17. Select **OK**.
18. Select **Save**.

The screenshot shows the SAP 'RFC Destination ISCRER' configuration window. The 'Connection Test' and 'Unicode Test' buttons are at the top. The 'RFC Destination' is 'ISCRER' and the 'Connection Type' is 'TCP/IP Connection'. The 'Description' field contains 'SAP Adapter RFC Listener for SAP System CER on Integration Server IS'. The 'Activation Type' is set to 'Registered Server Program'. The 'Registered Server Program' section shows 'Program ID' as 'ISCRER'. The 'Anstartensart des externen Programms' section has 'Gateway Standardwert' selected. The 'CPIC-Timeout' section has 'Gateway Standardwert' selected with a value of 60. The 'Gateway Options' section shows 'Gateway Host' as 'CERmain.wdf.sap.corp' and 'Gateway service' as 'sapgw21'. A 'Delete' button is next to the 'Gateway Host' field. A status bar at the bottom indicates 'Destination ISCRER saved'.

19. Remain on the current screen while you complete the steps for creating an Adapter for SAP 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 Adapter for SAP's administrative screens. For information about setting user privileges, see the *webMethods Integration Server Administrator's Guide* for your release.
2. Start Integration Server and Integration Server Administrator, if they are not already running.
3. Using Integration Server Administrator, make sure that the WmSAP package is enabled. To verify the status of the WmSAP package, see [“Enabling Packages” on page 47](#).
4. Using Designer, 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 45](#) for details.

Configuring an RFC Listener

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

➤ To create an RFC listener on Adapter for SAP

1. In the **Adapters** menu in the Integration Server Administrator navigation area, click **Adapter for SAP**.
2. In **Adapter for SAP** 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 Designer before you can specify it using this

Field	Description/Action
	<p>parameter. For general information about creating packages, see <i>webMethods Service Development Help</i> for your release.</p> <p>Note: Configure the listener in a user-defined package rather than in the adapter's package. See “Package Management” on page 45 for other important considerations when creating packages for Adapter for SAP.</p>
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". You can select secured ports beginning from "sapgw00s".
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 .
SNC Quality of Service	<p>SNC Quality of service, possible values:</p> <ul style="list-style-type: none"> ■ 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

Field	Description/Action
	<p>a service here that implements specification <code>pub.sap.listener:listenerAuthorizationCheck</code>. 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.</p> <p>See “pub.sap.listener:listenerAuthorizationCheck” on page 302 for information about the service specification.</p>
RFC Trace	<p>Whether you want RFC tracing enabled or disabled. For a production system, select Off. When you select On, Adapter for SAP collects the trace messages in <code>rfc*.trc</code> files in the directory <code>packages/WmSAP/logs</code>. The location can be changed with the <code>"watt.sap.jco.trace.dir"</code> configuration switch.</p> <p>For detailed information about logging, see Logging and Monitoring. For information about changing the directory for the trace files, see “Adapter Configuration” on page 249.</p>
Log transaction status	<p>This switch can be set to <code>On</code> or <code>Off</code>.</p> <p>If set to <code>Off</code>, 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.</p> <div> <p>Tip:</p> <p>Setting this switch to <code>Off</code> 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 Adapter for SAP.</p> </div>
Store message body	<p>This switch can be set to <code>On</code> or <code>Off</code>.</p> <p>If set to <code>Off</code>, 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.</p> <div> <p>Tip:</p> <p>Setting this switch to <code>Off</code> 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 Adapter for SAP.</p> </div>
<p>Important:</p> <p>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.</p>	

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 Integration Server 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 94](#) and [“Configuring Listener Notifications” on page 105](#).

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 Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listeners**. The Listeners screen appears.
3. Select **Enabled** from the list in the **State** field. 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 Adapter for SAP and the SAP system.

Tip:

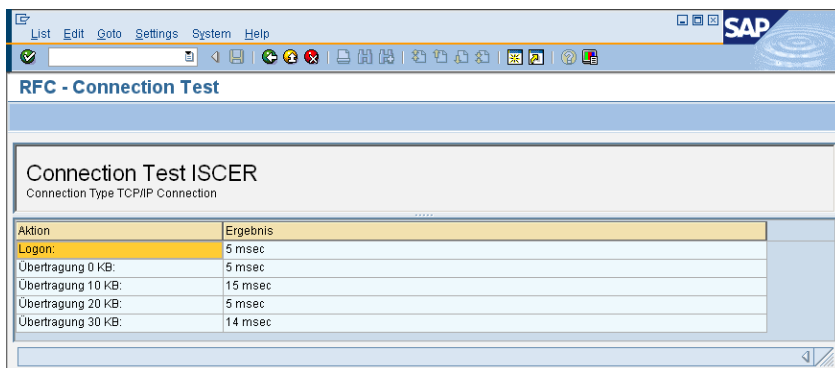
The **Enable all suspended** link helps you change the state quickly for multiple listeners.

Testing the RFC Listener

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

➤ 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 Adapter for SAP 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 Listeners


You can view listeners and each listener's parameters from Integration Server Administrator or from Designer. You can also view the notification order of a listener.

Viewing Listeners Using Integration Server Administrator

➤ To view listeners using Integration Server Administrator

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listeners**.

The Listeners screen appears, listing all of the current listeners. You can control the number of listeners that are displayed on this screen. For more information, see [“Controlling Pagination” on page 28](#).

3. To view a listener's parameters:
 - a. On the Listeners screen, click the  icon for the listener that you want to see.
 - b. The View Listener screen displays the parameters for the listener. For descriptions of the listener parameters, see [“Configuring an RFC Listener” on page 94](#).
4. Click **Return to Adapter for SAP Listeners** to return to the Listeners screen.

Sorting and Filtering Listeners Using Integration Server Administrator

➤ To sort and filter listeners using Integration Server Administrator

You can sort and filter the list of listeners that appears on the Listeners screen.

- To sort information on the Listeners screen, click the **Up** and **Down** arrows.
- To filter the list of listeners:
 1. On the Listeners screen, click **Filter Listeners**.
 2. Type the criterion by which you want to filter into the **Filter criteria** box. Filtering is based on the package name, not the listener name. To locate all listeners containing specific alphanumeric characters, use asterisks (*) as wildcards. For example, if you want to display all listeners containing the string "abc", type *abc* in the **Filter criteria** box.
 3. Click **Submit**. The Listeners screen displays the listeners that match the filter criteria.
 4. To re-display all listeners, click **Show All Listeners**.

Viewing Listeners Using Designer


➤ To view listeners using Designer

1. Start Designer if it is not already running.
2. From the Designer Package Navigator view, open the package and folder in which the listener is located.
3. Double-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 94](#).

Viewing the Notification Order of a Listener

➤ To view the notification order of a listener

1. In the **Adapters** menu in the navigation area of the Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listeners**.
3. On the Listeners screen, click the  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, see [“Editing the Notification Order of a Listener” on page 100](#).

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


Editing Listeners

You use Integration Server 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 [“Editing the Notification Order of a Listener” on page 100](#).

Editing a Listener

➤ To edit a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** 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 103](#).
4. On the Listeners screen, click the  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 94](#).

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

Editing the Notification Order of a Listener

➤ To edit the notification order of a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** 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 103](#) for details.
4. On the Listeners screen, click the  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 Adapter for SAP 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 104](#).

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 Integration Server Administrator.

➤ To copy a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listeners**.
3. On the Listeners screen, click the  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 94](#).

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 Integration Server Administrator to delete listeners.

Important:

If you delete Adapter for SAP 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 100](#).

➤ To delete a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** 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 103](#) for details.
4. On the Listeners screen, click the  icon for the listener you want to delete.

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 Adapter for SAP has the same effect as disabling them. For more information about disabling listeners, see [“Disabling Listeners” on page 103](#).

➤ To suspend a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listeners**.

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

> To disable a listener

1. In the **Adapters** menu in the navigation area of Integration Server Administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listeners**.

The Listeners screen appears.

3. Select **Disabled** from the list in the **State** field. 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 Adapter for SAP listener notifications. Adapter for SAP has the following types of listener notifications that you can configure:

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

For more information on how listener notifications work, see [“Adapter Listeners and Listener Notifications” on page 26](#).

Before You Configure Listener Notifications

➤ To prepare to configure a listener notification

1. Install webMethods Integration Server and Adapter for SAP on the same machine. For details, see [“Installing, Upgrading, and Uninstalling webMethods Adapter for SAP 10.1” on page 37](#).
2. Make sure that you have webMethods administrator privileges so that you can access Adapter for SAP's administrative screens. For more information about setting user privileges, see *webMethods Integration Server Administrator's Guide* for your release.
3. Start Integration Server and Integration Server Administrator, if they are not already running.
4. Using Integration Server Administrator, make sure that the WmSAP package is enabled. To verify the status of the WmSAP package, see [“Enabling Packages” on page 47](#).
5. Configure a listener using Integration Server Administrator. For more information on how to configure a new listener, see [“Configuring an RFC Listener” on page 94](#).
6. Using Designer, 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 45](#).

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 58 .	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 91 .	SAP system
3. Configure an RFC listener. For details, see “Configuring an RFC Listener” on page 94 .	Integration Server Administrator
4. Select the appropriate notification template and configure the notification. For instructions to configure notifications, see “Configuring Listener Notifications” on page 105 .	Designer

Task	Use this tool...
5. For synchronous publish-and-wait and for asynchronous messaging, you should create an Integration Server trigger that subscribes to the document type that Adapter for SAP created with the notification. Refer to <i>webMethods Service Development Help</i> for more information about using triggers.	Designer
6. Enable the adapter notifications. For instructions to enable listener notifications, see “Enabling Listener Notifications” on page 115 .	Integration Server Administrator

Configuring Listener Notifications

Note:

Routing notifications are used when no RFC listener notification has been defined. For more information about routing listeners, see [“Routing Messages Through Adapter for SAP” on page 139](#).

RFC Listener Notification (Synchronous)

This section describes how to create and configure a synchronous RFC listener notification.

Creating a Synchronous RFC Listener Notification

➤ To create a synchronous RFC listener notification

1. Start Software AG Designer.
2. Select **New > Adapter Notification** and do one of the following:
 - If you are using Designer, be sure the parent namespace is selected and type a name for the adapter notification. Click **Next**.
3. Select **Adapter for SAP** as the adapter type and click **Next**.
4. From the list of available templates, select **RFC Listener Notification (synchronous)** and click **Next**.
5. Select the appropriate **Notification Listener Name** and click **Next**.
6. Select a service and click **Next**.
7. Click **Finish**.

The Adapter Notification template creates the following items:

- An RFC synchronous listener notification
- Two synchronous document types: a synchronous reply document type and a 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. Integration Server does not publish these document types.

Configuring a Synchronous RFC Listener Notification

➤ To configure a synchronous RFC listener notification

1. In the adapter notification service editor for the notification you just created, select the **Adapter Settings** tab.
2. In the Adapter Properties area, confirm the adapter name, adapter listener name, and adapter notification template.
3. In the Execution Mode area, do one of the following to specify an execution mode:
 - To specify a flow service to invoke directly:
 1. Select **Service Invoke**.

This option invokes a local flow service directly, returns values from the service as non-publishable documents, and is the default.
 2. Click the Browse button to navigate to and select a service.
 - To publish documents locally or to a Broker and wait for a reply:
 1. Select **Publish and Wait**.

This option publishes the request either to the local Integration Server or to the webMethods Broker connected to that Integration Server and waits for a reply.
 2. Select **true** to publish documents locally (to this Integration Server only) or **false** to publish to the Broker connected to this Integration Server.

Note that if no Broker is configured for Integration Server, documents will be published locally.
 3. Specify the number of milliseconds to wait for a reply. The default is -1, which means to wait indefinitely for a reply.
 4. Create a matching Broker/local trigger to process the request document locally or at a remote Integration Server connected to this Integration Server by way of a Broker. Configure the trigger to ensure that the matching reply document is returned to this

adapter notification. For more information about this step, see *webMethods Service Development Help* for your release.





4. 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.

5. Select **Request Field Selection** tab to specify which RFC parameters to include in the request document. The request fields are available for mapping steps at design time.

Note:

Irrespective of the selection you make at design time, the adapter makes all fields available in the pipeline at run time.

6. In the **Use** column on the **Request Field Selection** tab, select the boxes for the required request fields. You can also select all fields by using the  icon or unselect all fields by using the  icon.
7. Select the **Reply Field Selection** tab to specify which fields should match the returning message from the notification.
8. In the **Use** column on the **Reply Field Selection** tab, select the boxes for the required reply fields. You can also select all fields by using the  icon or unselect all fields by using the  icon.
9. 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 89](#).

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 Service Development Help* for your release.
11. From the **File** menu, select **Save** (or **Save All**).

ALE Listener Notification (Synchronous)

This section describes how to create and configure a synchronous ALE listener notification.

Creating a Synchronous ALE Listener Notification

➤ To create a synchronous ALE listener notification

1. Start Software AG Designer.
2. Select **New > Adapter Notification** and do one of the following:
 - If you are using Designer, be sure the parent namespace is selected and type a name for the adapter notification. Click **Next**.
3. Select **Adapter for SAP** as the adapter type and click **Next**.
4. From the list of available templates, select **ALE Listener Notification (synchronous)** and click **Next**.
5. Select the appropriate **Notification Listener Name** and click **Next**.
6. Select a service and click **Next**.
7. Click **Finish**.

The Adapter Notification template creates the following items:

- An ALE synchronous listener notification
- Two synchronous document types: a synchronous reply document type and a 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.

Configuring a Synchronous ALE Listener Notification

> To configure a synchronous ALE listener notification

1. In the adapter notification service editor for the notification you just created, select the **Adapter Settings** tab.
2. In the Adapter Properties area, confirm the adapter name, adapter listener name, and adapter notification template.
3. In the Execution Mode area, do one of the following to specify an execution mode:
 - To specify a flow service to invoke directly:
 1. Select **Service Invoke**.
This option invokes a local flow service directly, returns values from the service as non-publishable documents, and is the default.
 2. Click the Browse button to navigate to and select a service.
 - To publish documents locally or to a Broker and wait for a reply:
 1. Select **Publish and Wait**.
 2. This option publishes the request either to the local Integration Server or to the webMethods Broker connected to that Integration Server and waits for a reply.
 3. Select **true** to publish documents locally (to this Integration Server only) or **false** to publish to the Broker connected to this Integration Server.
Note that if no Broker is configured for the Integration Server, documents will be published locally.
 4. Specify the number of milliseconds to wait for a reply. The default is -1, which means to wait indefinitely.
 5. Create a matching Broker/local trigger to process the request document locally or at a remote Integration Server connected to this Integration Server by way of a Broker. Configure the trigger to ensure that the matching reply document is returned to this adapter notification. For more information about this step, see the *webMethods Service Development Help* for your release.
4. 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).

Property	Description
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 Adapter for SAP 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.

5. Select the **Request Field Selection** tab to specify which IDoc fields to include in the request document. The request fields are available for mapping steps at design time.

Note:

By default, irrespective of the selection you make at design time, the adapter makes all fields available in the pipeline at run time.

6. In the **Use** column on the **Request Field Selection** tab, select the boxes for the required request fields. You can also select all fields by using the  icon or unselect all fields by using the  icon.
7. 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 89](#).
8. 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 Service Development Help* for your release.
9. From the **File** menu, select **Save** (or **Save All**).

Editing Synchronous Listener Notification Services

If you specified Service Invoke as the execution mode of the listener notification when you first configured the synchronous listener notification, you may later need to change the service that is invoked. To change the notification, 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 119](#).

➤ To edit the notification service

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

2. In the Execution Mode area, click the **Browse** button next to the service.
3. Locate and select a new service.
4. Click **OK**.
5. From the **File** menu, select **Save** (or **Save All**).

RFC Listener Notification (Asynchronous)

This section describes how to create and configure an asynchronous RFC listener notification.

Creating an Asynchronous RFC Listener Notification

➤ To create an asynchronous RFC listener notification

1. Start Software AG Designer.
2. Select **New > Adapter Notification** and do one of the following:
 - If you are using Designer, be sure the parent namespace is selected and type a name for the adapter notification. Click **Next**.
3. Select **Adapter for SAP** as the adapter type and click **Next**.
4. From the list of templates, select **RFC Listener Notification (asynchronous)** and click **Next**.
5. Select the appropriate **Notification Listener Name** and click **Next**.
6. Click **Finish**.

The Adapter Notification template creates the following items:

- An RFC asynchronous listener notification
- A Publish Document Type

Configuring an Asynchronous RFC Listener Notification

➤ To configure an asynchronous RFC listener notification

1. In the adapter notification service editor for the notification you just created, select the **Adapter Settings** tab.
2. In the Adapter Properties area, confirm the adapter name, adapter notification name, and adapter notification template.

3. In the Publish Document to area, do one of the following:

■ To publish to the Broker connected to the local Integration Server:

1. Select **webMethods Broker/Local**.

This option publishes documents to the local Integration Server or to the Broker connected to that Integration Server, if one is configured. This is the default.

■ To publish to JMS:

1. Select **JMS Provider**.

This option publishes documents in the form of messages to a JMS provider.

2. Click the **Browse** button next to **Connection alias name**.3. Select the name of the connection alias as it is configured on Integration Server and then click **OK**.

If the connection alias is a Broker Cluster configured with Multisend Guaranteed policy, you must add the watt property `watt.art.notification.jmsSend.usePublicService` and set it to **true**.

Adapter Runtime does not support `LOCAL_TRANSACTION` and `XA_TRANSACTION` type JMS connection alias.

4. Type the destination name defined in the JMS provider to specify the target of messages the client produces and the source of messages it consumes.

5. Specify whether the destination is a **Queue** or a **Topic**. The default is Queue.

4. Create a matching Broker/local or JMS trigger to process the request document locally, at a remote Integration Server connected to this Integration Server by way of a Broker, or to a JMS provider, depending on the "publish to" destination you selected in the previous step. Configure the trigger to ensure that the matching reply document is returned to this adapter notification.

5. 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.

Property	Description
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.

6. Select the **Request Field Selection** tab to specify which RFC parameters to include in the publishable document. The request fields are available for mapping steps at design time.

Note:

Irrespective of the selection you make at design time, the adapter publishes all fields available to the Broker at run time.

7. In the **Use** column on the **Request Field Selection** tab, select the boxes for the required request fields. You can also select all fields by using the  icon or unselect all fields by using the  icon.
8. 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 Service Development Help* for your release.
9. From the **File** menu, select **Save** (or **Save All**).

ALE Listener Notification (Asynchronous)

This section describes how to create and configure an asynchronous ALE listener notification.

Creating an Asynchronous ALE Listener Notification

➤ To create an asynchronous ALE listener notification

1. Start Software AG Designer.
2. Select **New > Adapter Notifications** and do one of the following:
 - If you are using Designer, be sure the parent namespace is selected and type a name for the adapter notification. Click **Next**.
3. Select **Adapter for SAP** as the adapter type and click **Next**.
4. From the list of templates, select **ALE Listener Notification (asynchronous)** and click **Next**.
5. Select the appropriate **Notification Listener Name** and click **Next**.

6. Click **Finish**.

The Adapter Notification template creates the following items:

- An ALE asynchronous listener notification
- A Publish Document Type

Configuring an Asynchronous ALE Listener Notification

> To configure an asynchronous ALE listener notification

1. In the adapter notification service editor for the notification you just created, select the **Adapter Settings** tab.
2. In the Adapter Properties area, confirm the adapter name, adapter notification name, and adapter notification template.
3. In the Publish Document to area, do one of the following:

- To publish to the Broker connected to the local Integration Server:

1. Select **webMethods Broker/Local**.

This option publishes documents to the local Integration Server or to the Broker connected to that Integration Server, if one is configured. This is the default.

- To publish to JMS:

1. Select **JMS Provider**.

This option publishes documents in the form of messages to a JMS provider.

2. Click the **Browse** button next to **Connection alias name**.
3. Select the name of the connection alias as it is configured on Integration Server and then click **OK**.

If the connection alias is a Broker Cluster configured with Multisend Guaranteed policy, you must add the watt property `watt.art.notification.jmsSend.usePublicService` and set it to **true**.

Adapter Runtime does not support LOCAL_TRANSACTION and XA_TRANSACTION type JMS connection alias.

4. Type the destination name defined in the JMS provider to specify the target of messages the client produces and the source of messages it consumes.
 5. Specify whether the destination is a **Queue** or a **Topic**. The default is Queue.
4. Create a matching Broker/local or JMS trigger to process the request document locally, at a remote Integration Server connected to this Integration Server by way of a Broker, or to a JMS

provider, depending on the "publish to" destination you selected in the previous step. Configure the trigger to ensure that the matching reply document is returned to this adapter notification.



5. 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 Adapter for SAP 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.

6. Select the **Request Field Selection** tab to specify which IDoc fields to include in the publishable document. The request fields are available for mapping steps at design time.

Note:

By default, irrespective of the selection you make at design time, the adapter publishes all fields available to the Broker at run time.

7. In the **Use** column on the **Request Field Selection** tab, select the boxes for the required request fields. You can also select all fields by using the  icon or unselect all fields by using the  icon.
8. 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 Service Development Help* for your release.
9. From the **File** menu, select **Save** (or **Save All**).

Enabling Listener Notifications

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

> To enable a listener notification

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

2. In **Adapter for SAP** menu, select **Listener Notifications**.
3. On the Listener Notifications screen, click **No** in the **Enabled** column for the listener notification you want to enable.

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 Integration Server Administrator. For instructions to configure a listener, see [“Configuring an RFC Listener” on page 94](#).
2. Configure a listener notification using Designer. For instructions to configure a notification, see [“Configuring Listener Notifications” on page 105](#).
3. Enable the listener notification using Integration Server Administrator. For instructions to enable a listener notification, see [“Enabling Listener Notifications” on page 115](#).
4. Enable the listener using Integration Server Administrator. For instructions to enable a listener, see [“Enabling Listeners” on page 97](#).
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 in Designer . When you test a publishable document type, you provide input values that Designer 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). Designer 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:

In Designer, prior to running the PublishDocument, first uncheck the **Field must exist at run-time** field on the msgBody property. To access the field, right-click **msgBody**, select **Properties**, and then select the **Constraints** tab. When you test a publishable document type, Integration Server actually publishes the document locally or to Broker (whichever is specified).

For instructions to test a publishable document type, see the *webMethods Service Development Help* for your release. Also, for a complete description of the envelope parameters located in the WmPublic folder, see the *webMethods Integration Server Built-In Services Reference*. 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 can view listener notifications from Integration Server Administrator, and Designer.

Viewing Listener Notifications Using Integration Server Administrator

➤ To view listener notifications using Integration Server Administrator

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

The Listener Notifications screen appears, listing all the listener notifications. You can control the number of listener notifications that are displayed on this screen. For more information, see [“Controlling Pagination” on page 28](#).

2. In **Adapter for SAP** menu, click **Listener Notifications**.

Sorting and Filtering Listener Notifications

➤ To sort and filter listener notifications

You can sort and filter the list of listener notifications that appears on the Listener Notifications screen.

- To sort information on the Listener Notifications screen, click the **Up** and **Down** arrows.
- To filter the list of listener notifications:
 1. On the Listener Notifications screen, click **Filter Listener Notifications**.
 2. Type the criterion by which you want to filter into the **Filter criteria** box. Filtering is based on the notification name. To locate all listener notifications containing specific alphanumeric characters, use asterisks (*) as wildcards. For example, if you want to display all notifications containing the string "abc", type *abc* in the **Filter criteria** box.
 3. Click **Submit**. The Listener Notifications screen displays the listener notifications that match the filter criteria.
 4. To re-display all listener notifications, click **Show All Listener Notifications**.

Viewing Listener Notifications Using Designer

➤ To view listener notifications using Designer

1. In Designer, 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 adapter's Adapter Notification Editor displays details about the configured listener notification.

Editing Listener Notifications

You use Designer to edit both synchronous and asynchronous listener notifications. 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. Listener notifications must be disabled before you can edit or delete them.

➤ To edit a listener notification

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

The adapter's Adapter Notification Editor displays details about the configured listener notification.

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 105](#) 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.

Editing Document Types Used by the Listener

➤ To edit the document types used by the listener

1. In Designer, 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 Adapter for SAP listener notification, you can delete it by following the instructions in this section. You delete listener notifications, both synchronous and asynchronous, using Designer. Listener notifications must be disabled before you can edit or delete them.

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 Designer, 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 Integration Server Administrator.

➤ To disable a listener notification

1. In the **Adapters** menu in the IS Administrator navigation area, click **Adapter for SAP**.
2. In **Adapter for SAP** 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 Integration Server. It describes an application in which the SAP system requests a service to retrieve information about a product.

Note:

An Adapter for SAP 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.

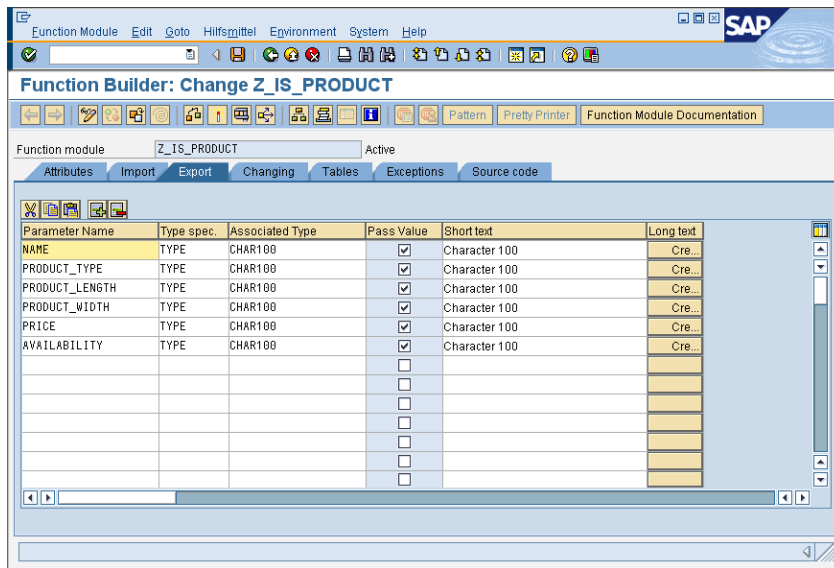
Note:

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 Adapter for SAP is the field **Processing type**. Select **Remote-Enabled Module** to allow this function to call externally to Adapter for SAP.
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 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 Designer and select **File > New > Adapter Notification**.
2. Select the destination directory and enter the name of the new notification. Click **Next**.
3. Select **Adapter for SAP** from the list of available adapter types. Click **Next**.
4. From the list of available templates, select **RFC Listener Notification**. Click **Next**.
5. 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**.
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 Adapter for SAP to search for several RFCs with similar names.

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

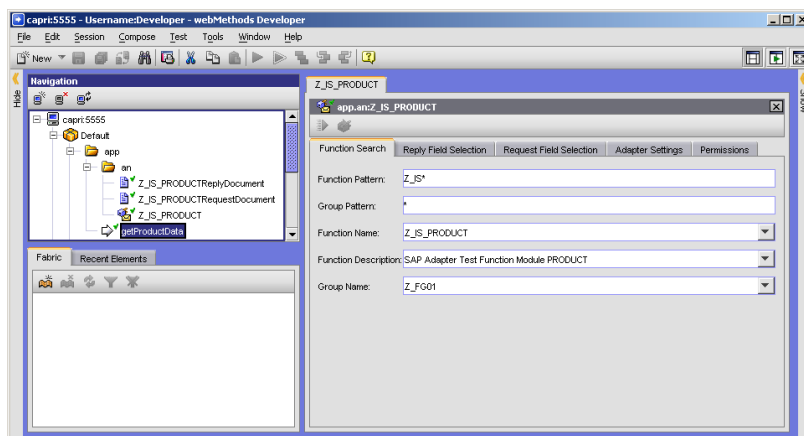
A list of RFC names that match the criteria is displayed.

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 58.

9. Click **Save** 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 **Adapter for SAP**.
 - b. In Adapter for SAP menu, click **Listeners**.

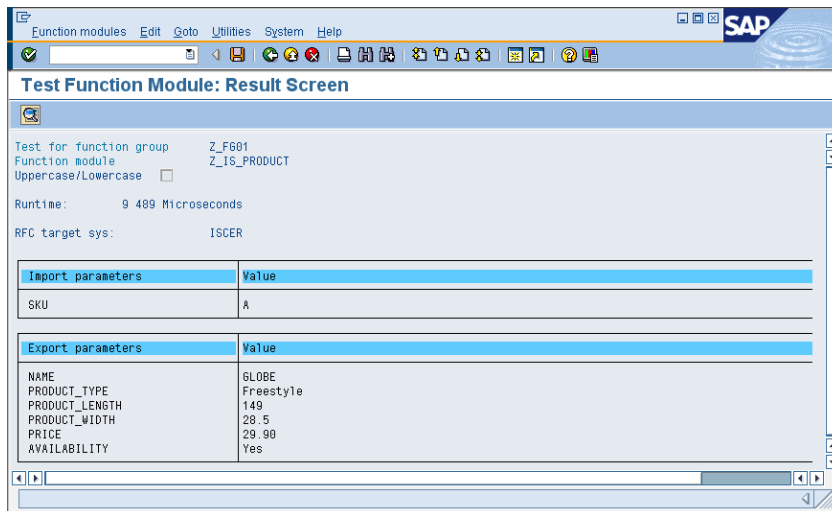
- c. Verify that the RFC listener you created for the RFC destination on the SAP system hosting Z_IS_PRODUCT shows status "Enabled", indicating that the listener is started and active.

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 94](#).

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 Adapter for SAP, 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” on page 124](#)
2. [“Create an Asynchronous ALE Listener Notification for IDoc Type Orders02” on page 124](#)
3. [“Define the Input and Output Signatures for the mapOrders02 Service” on page 125](#)
4. [“Map Fields from Orders02RequestDocument to PurchaseOrder Document” on page 125](#)
5. [“Testing the Listener Notification from the SAP System” on page 125](#)

Create an Empty Flow Service Called mapOrders02

Use the following procedure to create a flow service named app.idocs:mapOrders02 with Designer (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 Designer, select the package that contains the ALE notification 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 **Adapter for SAP** 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 124](#), 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  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 126](#).

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

➤ 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.
4. 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 Adapter for SAP, use the SAPGui to perform the following steps:

1. [“Create an RFC Destination” on page 126](#)
2. [“Define a Logical Port” on page 126](#)
3. [“Create a Partner” on page 127](#)
4. [“Create a Partner Profile” on page 127](#)
5. [“Create a Distribution Model for the Partner and Message Type with SAP System 4.5 or Earlier” on page 128](#)

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 91](#).

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.

To define a logical port, use **WE21**, or alternatively, the following procedure:

➤ To define a logical port

1. In the Main screen, select **Tools > Business Communication > IDoc-Basis > IDoc > Port Definition**.
2. Select the **Transactional RFC** tree item and click **Create**.

3. On the toolbar, click **New Entries**.
4. Either select your own descriptive port name or let the system generate one.
5. 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. To create a partner (logical system), use **SPRO_ADMIN**, or alternatively, the following procedure:

➤ To create a partner (logical system)

1. In the Main screen, select **Tools > AcceleratedSAP > Customizing > Project Management**.
2. Select **SAP Reference IMG**.
3. 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).
4. Select **Define Logical System**.
5. Click **New Entries**.
6. 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

To create a partner profile, you can use **WE20** or alternatively, the following procedure or alternatively, the following procedure:

➤ To create a partner profile

1. In the Main screen, select **Tools > Business Communication > IDoc-Basis > IDoc > Partner profile**.
2. Select the **LS (logical system) partner** type in the tree view and click **Create**.
3. Enter in the **Partner** field the partner you created in [“Create a Partner” on page 127](#), and save the partner profile.

4. Below the outbound parameter table control, click **Insert entry**.
5. Enter the message type of the IDoc, (for example: MATMAS).
6. Enter the logical receiver port you created before and enter the basic type of the IDoc, (for example: MATMAS03).
7. Save the outbound parameter.
8. Below the inbound parameter table control, click **Insert entry**.
9. Enter the message type of the IDoc, (for example: MATMAS) and the process code, (for example: MATM).
10. Save the inbound parameter.

Create a Distribution Model for the Partner and Message Type with SAP System 4.5 or Earlier

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

➤ To use SAP system 4.5 or earlier to create a distribution model

1. In the Main screen, select **Tools > Business Framework > ALE > Customizing**.
2. Open the **Cross-Application Components** folder, then the **Distribution (ALE)** folder, then the **Distribution Customer Model** folder in the tree view.
3. Select **Maintain customer distribution model**.
4. Create a new model using **Model > Create**.
5. 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).

Create a Distribution Model for the Partner and Message Type with SAP System 4.6 or Later

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

➤ To use SAP system 4.6 or later to create a distribution model

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).

9 Generating Document Types

■	Generating Document Types for RFC Structure	132
■	Generating Document Types for IDocs	132

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 Adapter for SAP plug-in installed on the machine that hosts Designer .

➤ **To create a document type for RFC Structure**

1. In Designer, select **File > New > Document Type**.
2. Select the parent namespace and name for the document type, and click **Next**.
3. From the list of source files, select **SAP** as the type of source file from which to create the document type. Click **Next**.
4. Select **RFC** as the **SAP Document Type** and click **Next**.
5. Select the **SAP System ID** and click **Next**.
6. Select the **RFC Structure**. If no structures have been cached, you must type the structure name into the field.
7. Click **Next** and then **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 Adapter for SAP plug-in installed on the machine that hosts Designer .

➤ **To create a document type using the DDIC**

1. In Designer, select **File > New > Document Type**.
2. Select the parent namespace and name for the document type, and click **Next**.

3. From the list of source files, select **SAP** as the type of source file from which to create the document type. Click **Next**.
4. Select **IDoc** as the **SAP Document Type** and click **Next**.
5. Select the **SAP System ID** and click **Next**.
6. Fill in the following fields:

Field	Description/Action
IDoc Type	The IDoc version.
CIM Type	Optional. The IDoc type extension (CIM type/customer extension type).
SAP System Release	Optional. The IDoc release.
Old IDoc Type 2	Optional. The IDoc version. Uncheck for a new version 3 IDoc, check for old version 2 IDocs (as in 3.1 SAP systems).

7. Click **Finish**.

Generating an IDoc Document Type from an XML Document

➤ To create an IDoc document type from an XML document

1. In Designer, select **File > New > Document Type**.
2. Select the parent namespace and name for the document type. Click **Next**.
3. From the list of source files, select **XML**. Click **Next**.
4. To select the source location, do one of the following in the **File/URL** field:
 - To create the document type from an XML document on your local file system, type the path and file name, or click the **Browse** button to navigate to and select the file.
 - To create the document type from an XML document that resides on the Internet, type the URL of the resource. The URL you specify must begin with http: or https:
5. Click **Finish**.

Important:

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.

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. In Designer, select **File > New > Document Type**.
2. Select the parent namespace and name for the document type. Click **Next**.
3. From the list of source files, select **DTD**. Click **Next**.
4. To select the source location, do one of the following in the **File/URL** field:
 - To create the document type from a DTD on your local file system, type in the path and file name, or click the **Browse** button to navigate to and select the file.
 - To create the document type from a DTD that resides on the Internet, type the URL of the resource. The URL you specify must begin with http: or https:
5. Click **Next**.
6. Under **Select the root node**, select the root element of the DTD.
7. Under **Element reference handling**, select one of the following:
 - Select **Only generate document types for elements with multiple references** to instruct Integration Server to create a separate document type for a referenced element only when the DTD contains multiple references to that element.
 - If an element is referenced multiple times, Integration Server creates a separate document type for the element. Integration Server replaces each element reference with a document reference field.
 - If an element is referenced only once, Integration Server defines the element in line by replacing the element reference with a document field.
 - Select **Always generate document types for referenced elements** to instruct Integration Server to always create a separate document type for a referenced element even if it is referenced only once. In the document type, Integration Server replaces each element reference with a document reference field.
8. Click **Finish**.

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 Designer 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 generate a document type from a sample IDoc, in Designer you must:

1. Save a pipeline image capturing the IDoc.
2. Retrieve the created pipeline image.


Saving a Pipeline Image Capturing the IDoc

➤ To save a pipeline image capturing the IDoc

1. Open Designer.
2. On the **File** menu, select **New**.
3. On the **New** panel, select **Flow Service**.
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 **Element 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, for example `app:mapOrders02`.)
5. Click **Finish**.
6. Now switch to the empty flow service you just created.
7. Click the ➡ icon on the **Flow Pane** toolbar and select the `pub.flow:savePipeline` service. If the service does not appear in the list, select **Invoke** to find it.

The service will copy the contents of the pipeline so that you can retrieve the pipeline at 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 the ↺ icon on the toolbar.

10. Type a name for the saved pipeline and select **OK**.
11. Click the  icon to save the flow service.
12. To send your sample IDoc to the service:
 - Use the SAGGui or the utility at `/WmSAP/submitIDocXML.html`
 - 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 145](#).


When the routing listener receives the document that you submit, 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.

Retrieving the Created Pipeline Image

➤ To retrieve the pipeline image you created

1. In Designer, create a new flow service in the same way you created `app.idocs:mapOrders02`.
2. Select the **Flow** tab.
3. Select the  icon on the **Flow Pane** toolbar and select the `pub.flow:restorePipeline` service. If the service does not appear in the list, select **Invoke** to locate it. The service will retrieve the contents of the pipeline you saved previously.
4. Set the **\$name** input field of the service to the name of the saved pipeline and select **OK**.
5. As the next step of the service, insert:

If IDoc sent to Adapter for SAP	Service Name	Service Location
over http	<code>pub.xml.xmlNodeToDocument</code>	<code>WmPublicPackage</code>
over tRFC from an SAP System	<code>pub.sap.idoc:DocToDocument</code>	<code>WmSAPPackage</code>
from the sample page <code>/WmSAP/submitIDocXMLhtml</code>	<code>pub.sap.idoc:DocToDocument</code>	<code>WmSAPPackage</code>

6. Click the  icon to save the flow service.
7. Execute the service and then select the **Results** tab and locate the *document* variable. It should contain the data of your IDoc.
8. 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. 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 the  icon to save the document type.

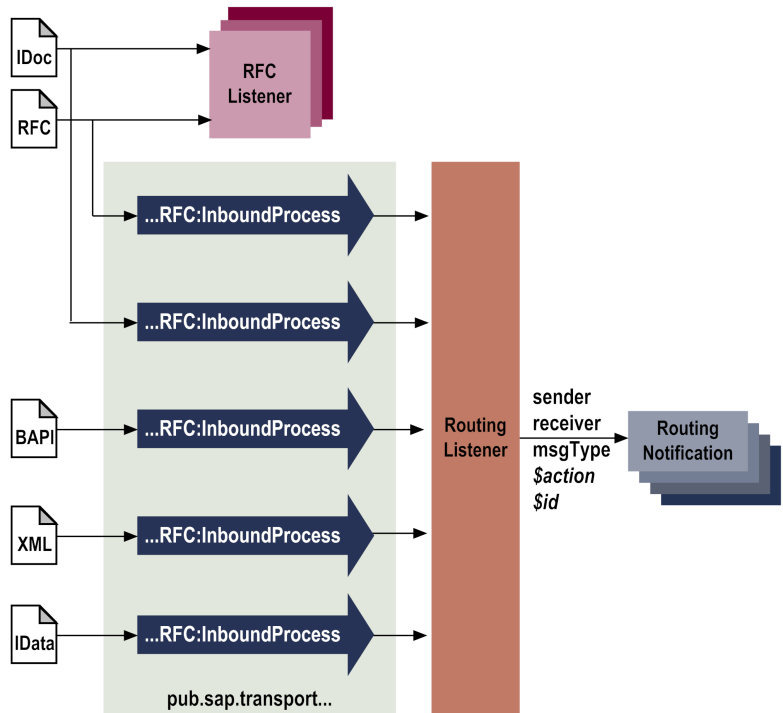
10 Routing Messages Through Adapter for SAP

■ Introduction	140
■ Overview	141
■ Routing Notifications	145
■ Sending an RFC from an SAP System to Adapter for SAP	150
■ Sending a BAPI from an SAP System to Adapter for SAP	155
■ Sending IDocs with ALE from an SAP System to Adapter for SAP	158
■ Routing RFCs Through Adapter for SAP	159
■ Routing BAPIs Through Adapter for SAP	160
■ Routing IDocs Through Adapter for SAP	165
■ Mapping IDocs to Other Formats	167
■ Content-Based Routing and Mapping for IDocs	172
■ Routing Arbitrary XML Documents Through the > Routing/Mapping	174

Introduction

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

When Adapter for SAP 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 Adapter for SAP, see...
RFCs	“Sending an RFC from an SAP System to Adapter for SAP ” on page 150 and “Routing RFCs Through Adapter for SAP ” on page 159
BAPIs	“Sending a BAPI from an SAP System to Adapter for SAP ” on page 155 and “Routing BAPIs Through Adapter for SAP ” on page 160.

Route...	Through Adapter for SAP, see...
IDOCs	“Sending IDocs with ALE from an SAP System to Adapter for SAP ” on page 158 and “Routing IDocs Through Adapter for SAP ” on page 165.
XML	“Routing Arbitrary XML Documents Through the > Routing/Mapping” on page 174

Overview

Normally, the RFCs and IDocs sent to Adapter for SAP 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.

Adapter for SAP 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 Adapter for SAP 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, Adapter for SAP 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:

- Sender, receiver, and message type.
- A flag that indicates whether the `Confirm` event should be forwarded to the receiver.
- 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).
- How to route the message; that is, the transport.
- 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 Adapter for SAP.

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 > Adapter for SAP > 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 200](#) and [“Built-in Services” on page 259](#) 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 89](#).

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

This achieves two things: The state in Adapter for SAP'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 Adapter for SAP notification. From the IS 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 145](#). For instructions on how to edit the notification order list, see [“Routing Notification Order” on page 143](#).

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.

Editing the Notification Order of a Listener

➤ To edit the notification order of a listener

1. In the **Adapters** menu in the navigation area of the administrator, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listeners**.
3. Select the routing listener (wm.sap.internal.ls:routingListener) and change the **State** to **Disabled**.
4. Click the  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 Adapter for SAP 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 Adapter for SAP 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 Adapter for SAP 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 and configure a routing notification.

➤ To create a routing notification

1. Start Software AG Designer.
2. Select **New >Adapter Notification** and do one of the following:
 - If you are using Designer, be sure the parent namespace is selected and type a name for the adapter notification. Click **Next**.
3. Select **Adapter for SAP** as the adapter type and click **Next**.
4. From the list of available templates, select **Routing Notification (synchronous)** and click **Next**.

5. Select the routing listener `wm.sap.internal.ls:routingListener`. Click **Next**.
6. Select the service that should be invoked by this routing notification. Click **Next** and then click **Finish**.
7. On the Transport Settings tab, type in values for fields **Sender**, **Receiver**, and **Message** 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.	“IS Transport” on page 146
ALE	Route an IDoc to an SAP system	“ALE Transport” on page 147
RFC	Route an RFC to an SAP system	“RFC Transport” on page 147
BAPI	Route a BAPI to an SAP system	“BAPI Transport” on page 147
XML	Post a bXML, IDoc- or RFCXML or any arbitrary XML document to a URL.	“XML Transport” on page 148

Note:

All but the IS transport will override the service you assigned to the routing notification in step 7.

9. Click **Save** in Designer 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 adapters for SAP using the IS transport, if the receiving Adapter for SAP 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 Adapter for SAP 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 58](#).

Set the Configure ALE Transport parameters as follows:

Key	Value
serverName	<p>Name of the SAP system to which you want to route the IDoc. Select an RFC connection alias from the drop down list.</p> <p>The drop down list contains the RFC connection aliases as defined on Adapter for SAP. For instructions on how to define an RFC connection to an SAP system, refer to “Configuring Adapter Connections” on page 58.</p>

RFC Transport

Use this transport to route the execution of an RFC function module to an SAP system by using RFC or tRFC.

Set the transport parameters as follows:

Key	Value
serverName	<p>Name of the SAP system to which you want to route the RFC. Select an RFC connection alias from the drop down list.</p> <p>The drop down list contains the RFC connection aliases as defined on Adapter for SAP. For instructions on how to define an RFC connection to an SAP system, refer to “Configuring Adapter Connections” on page 58.</p>

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	<p>Name of the SAP system to which you want to route the BAPI. Select an RFC connection alias from the drop down list.</p> <p>The drop down list contains the RFC connection aliases as defined on Adapter for SAP. For instructions on how to define an RFC connection to an SAP system, refer to “Configuring Adapter Connections” on page 58.</p>						
Processing restrictions	<p>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 “Using BizTalk Envelopes with Adapter for SAP” on page 321). 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:</p>						
	<table> <tr> <td>no restrictions</td><td>Caller may decide to send both synchronous and asynchronous calls.</td></tr> <tr> <td>synchronous only</td><td>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.</td></tr> <tr> <td>asynchronous only</td><td>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.</td></tr> </table>	no restrictions	Caller may decide to send both synchronous and asynchronous calls.	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.
no restrictions	Caller may decide to send both synchronous and asynchronous calls.						
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	<p>Select between the XML dialects SAP-XML, bXML, Values-XML, Arbitrary XML or SOAP XRFC (XRFC with SOAP envelope).</p> <p>If you select SAP-XML the content type is set to <code>application/x-sap.idoc</code>(respectively <code>.rfc</code>). Therefore the receiving server has to understand this content type. (This can be overridden using the following flag.)</p>

Key	Value
	If Arbitrary XML is selected, the transport expects the XML document as string in the variable <i>xmlData</i> .
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.)
httpUser	An optional parameter that allows you to specify a username for authentication on the remote Web system.
httpPassword	An optional parameter that allows you to specify a password for authentication on the remote Web system.

Tip:

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

When sending an RFC to a second Adapter for SAP 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 Adapter for SAP 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 **Adapter for SAP**.
2. In **Adapter for SAP** 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, Adapter for SAP 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 **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Listener Notifications**.
3. Select the routing notification you want to delete and click  in **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 Adapter for SAP

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 94](#).

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 91](#).

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	DType	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, 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 Adapter for SAP, 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 Adapter for SAP 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 Adapter for SAP 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 Adapter for SAP use a routing notification, include the following information in the header table:

Keys	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 Adapter for SAP), you can include `transport` information in the header table. The transport indicates where Adapter for SAP is to route the incoming RFC. When Adapter for SAP 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 Adapter for SAP 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 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.

Key	Value
valueScope	<p>Where you want Adapter for SAP to store the connection to the remote server.</p> <p>To save the connection in your own session, specify <code>SESSION</code>. Use <code>SESSION</code> when the work being performed requires state be maintained.</p> <p>To save the connection in a shared area, specify <code>GLOBAL</code>. Use <code>GLOBAL</code> when the work being performed is stateless.</p>

- To route the RFC to an SAP system, use the RFC transport.

Key	Value
transport	<p>RFC</p> <p>This value identifies the transport. Specify the value exactly as specified above.</p>
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	<p>XML</p> <p>This value identifies the transport. Specify the value exactly as specified above.</p>
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 <code>SAP-XML</code> if you want the RFC in an XML format that is compliant with the SAP XML specification. Specify <code>Values-XML</code> 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 159](#).

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 Adapter for SAP and invoke a routing notification to route the message. When testing the function module from the SAP Gui, 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 Adapter for SAP, the routing listener invokes the routing notification that corresponds to the sender CERCLNT800, receiver DELL, and message type Z_DEMO_COPY. To determine how Adapter for SAP 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 Adapter for SAP 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 Adapter for SAP, the header table gets interpreted directly to determine how to route the RFC. Adapter for SAP routes the message to the SAP system known as CERCLNT750 on Adapter for SAP using transport RFC. Note that CERCLNT750 must correspond to the connection alias of an RFC connection that is configured in Adapter for SAP.

Sending a BAPI from an SAP System to Adapter for SAP

SAP systems communicate with Adapter for SAP 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.

Adapter for SAP 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 as explained in [“Configuring an RFC Listener” on page 94](#).

➤ To create a routing notification

1. Open Designer and select **New > Adapter Notification**. Click **Next**.
2. Select **Adapter for SAP** 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 Adapter for SAP, 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.
bapiName	The name of the BAPI method, to which the call should be mapped. This value is case-sensitive.

Name	Value
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 'ISCR'
IMPORTING
  RETURN = returnCode
TABLES
  COMPANYCODE_LIST = companyCodes
  SBCHEADER = header
.
```

For a synchronous example, see [“Calling a BAPI Synchronously from SAP System” on page 203](#). For an asynchronous example, see [“Calling a BAPI Asynchronously from an SAP System” on page 206](#).

Sending IDocs with ALE from an SAP System to Adapter for SAP

The routing listener receives IDocs from an SAP system if the function module (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 Adapter for SAP 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 94](#).

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 167](#).

Routing RFCs Through Adapter for SAP

Posting RFC Based IFR-compatible XML Messages

Adapter for SAP 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 Adapter for SAP.

These XML messages must provide a BizTalk XML envelope (for further information, see [“Using BizTalk Envelopes with Adapter for SAP” on page 321](#)). 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 Adapter for SAP

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 Adapter for SAP, a corresponding routing notification should exist. If no entry exists, the message will not be executed and Adapter for SAP 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 Adapter for SAP. 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 Adapter for SAP.

```
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>0A125F1315B3D24B00000001E</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 Adapter for SAP in some cases the SAP system expects a `commit` command to execute the call.

➤ To commit the transaction directly over HTTP

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:

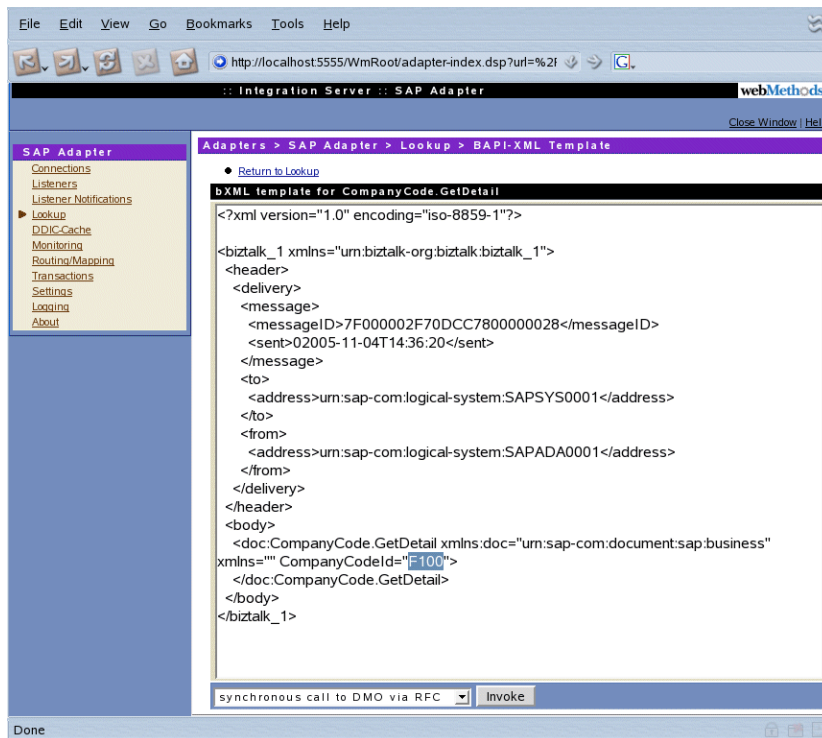
With the HTTP requests in steps - you need to resubmit the cookie that you got with the response in step 1. Otherwise, Integration Server cannot assign these requests to the same user session and the “commit” does not know which transaction to commit.

Synchronous Calls

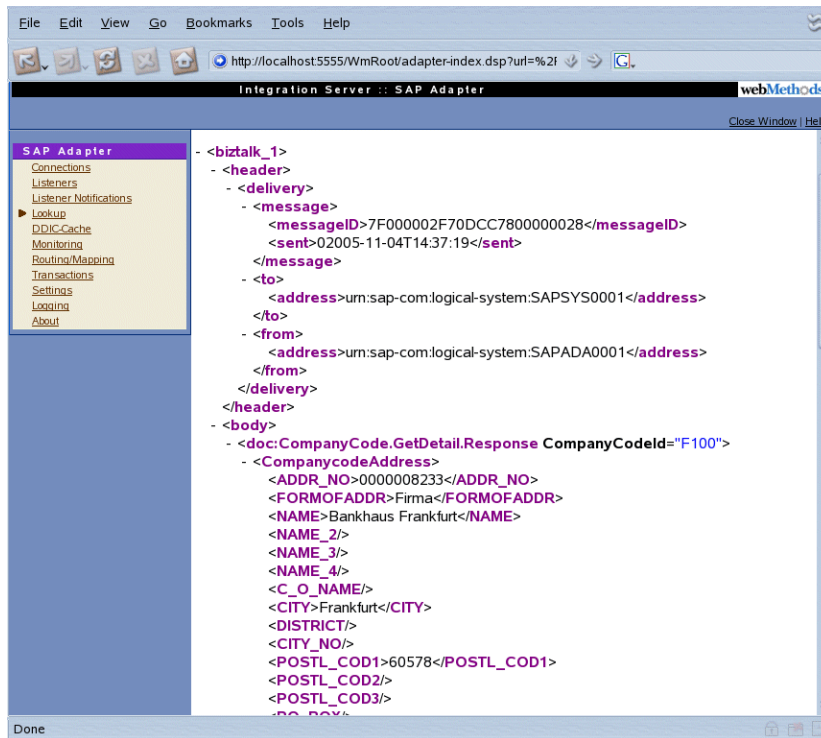
Application-level responses to synchronous XML requests are reported using a response business document in the HTTP response. The response business documents for synchronous calls are documented in the SAP Interface Repository and contain a serialization of all exporting and changing parameters of the BAPI or function module.

Example

- Sending a synchronous call to an SAP system



- It returns a response business document.



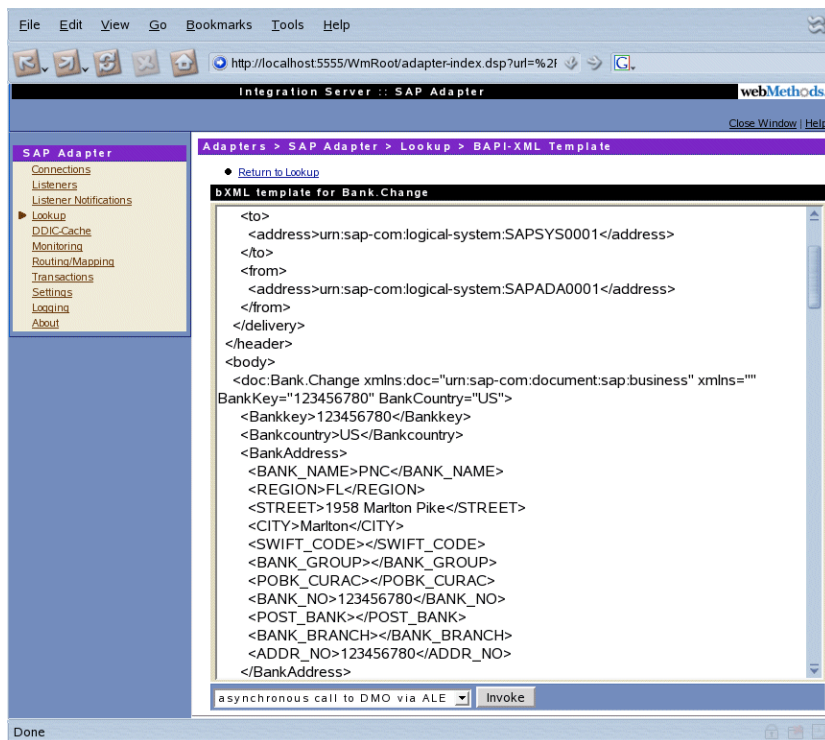
Asynchronous Calls

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.

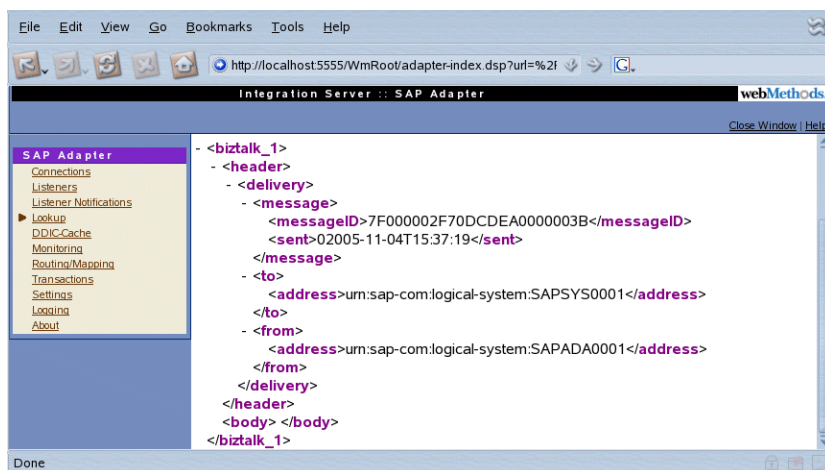
A transaction ID should be a global unique ID expressed in hexadecimal letters and 24 letters long, for example 0A11449F652C394A34DE042F. If an asynchronous request can be transmitted without errors inside the SAP system, Adapter for SAP will return a BizTalk envelope with an empty body as confirmation 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

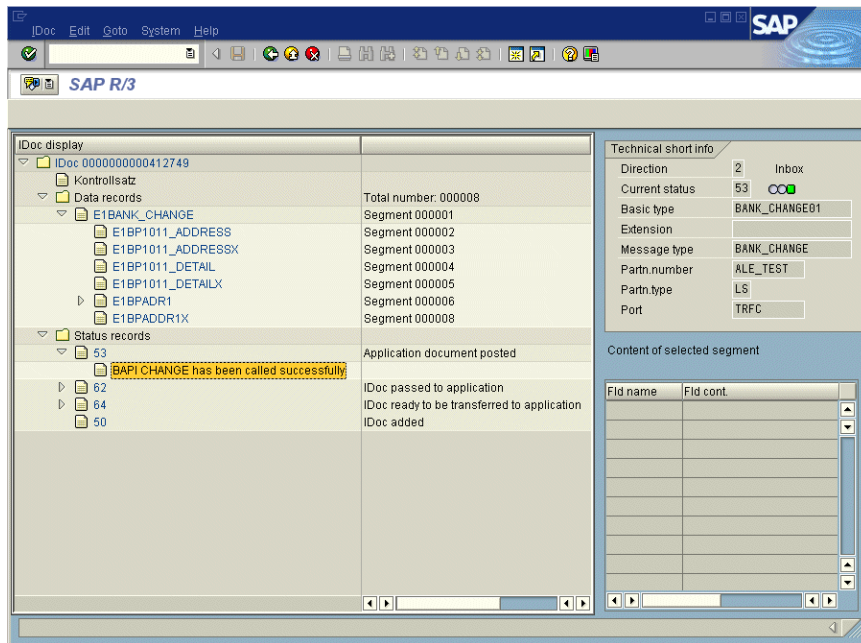
- Sending an asynchronous request to an SAP system by specifying a transaction ID



■ 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 Adapter for SAP

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.

➤ To simulate an HTTP client for testing purposes, you can use the service `pub.client:http`

1. Using Designer, 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 whether 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).

> To transmit IDocs between to Integration Servers

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* for your release. 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” on page 168](#)
2. [“Creating the Routing Notification” on page 168](#)
3. [“Creating a Document Type for Your IDoc” on page 168](#)
4. [“Transforming the IDoc to a Hierarchical Format” on page 169](#)
5. [“Mapping IDoc Information to Pipeline Variables” on page 169](#)
6. [“Testing the Mapping Service” on page 171](#)

Creating an Empty Flow Service

Use the following procedure to create a flow service named `app.idocs:mapOrders02` with Designer (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 from an SAP system you have an RFC connection to. See [“Create an IDoc Document Type Using the DDIC” on page 132](#) for more information.
- You can generate it from the DTD for the IDoc. See [“Generating an IDoc Document Type from a DTD” on page 134](#) for more information.

- You can generate it from a "sample" IDoc that you capture with Designer at design time. See [“Generating a Document Type from a Sample IDoc” on page 135](#) 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 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 Adapter for SAP through tRFC or through 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 the  icon on the Flow Pane toolbar and select the `pub.sap.idoc:iDocToDocumentService` 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 the  icon 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 Designer, select the **Flow** tab.

Note:

See *webMethods Service Development Help* for details.

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 the  icon 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 168](#), 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.

Defining Variables to which to Map Information from the IDoc

Perform the steps in the procedure below 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.
4. Click the  icon 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 *webMethods Service Development Help* for your release.

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 Adapter for SAP. 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 145](#).

Check the results of the service to ensure that the IDoc is being mapped correctly.

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

➤ To test the mapping service

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.

4. 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 Adapter for SAP 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 Adapter for SAP. For IDocs going to an SAP system, only mapping manipulation applies.

- **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 routing parameters 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 Adapter for SAP 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 > Adapter for SAP > 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 IS Administrator UI (**Adapters > Adapter for SAP > Routing/Mapping**).

Important:

In the SAP Routing/Mapping screen of the IS 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 IS 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 > Routing/Mapping

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...
<i>document</i>	This field contains the XML document as it would look like after having been processed by <code>pub.xml:xmlNodeToDocument</code> . Your service should extract the routing data from your specific documents.
<i>node</i>	Alternatively, you can provide your document as a node as used in the <code>pub.xml:xmlNodeToDocument</code> service as an input field.

Return Values

This key	Must specify...
<i>sender</i>	Sender used for finding the matching routing notification.
<i>receiver</i>	Receiver used for finding the matching routing notification.
<i>msgType</i>	Message type used for finding the matching routing notification.
<i>\$tid</i> (optional)	Transaction ID found in the document, if client wants to execute a transaction once and only once.
<i>\$action</i> (optional)	One of the following transaction codes: <ul style="list-style-type: none">■ 1 - Execute (default)■ 4 - Confirm

Configuring the Inbound Process Mechanism of the XML Transport

➤ To configure the Inbound Process mechanism of the XML transport

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 > Adapter for SAP > 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**:
5. Now you are ready to do routing using XML Inbound Process for arbitrary XML documents.

11 Transaction Handling

■ Managing Transactions and the Transaction Store	178
■ Viewing Transactions	178
■ Deleting Transactions	180
■ Automatic Cleanup of the Transaction Stores	181
■ Configuration Parameters for the Transaction Manager	182
■ Using the ALE Monitoring Features Via Adapter for SAP	183
■ Shared Transaction Store	188

Managing Transactions and the Transaction Store

Adapter for SAP comes with a transaction manager that allows you 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. If Adapter for SAP receives a message without a transaction ID, it can create one for the message.

The message itself will be stored into the transaction store. By default, the transaction store is located on Integration Server on which an individual Adapter for SAP is installed. Alternatively, it can be centrally located in an external directory in the file system shared by a group of Adapter for SAP or on an Integration Server on which one Adapter for SAP is designated as the server, and is available to a group of adapters for SAP. For complete information about configuring and working with the Shared Transaction Store, see [“Shared Transaction Store” on page 188](#).

Use the Transactions screens to view and manage the transactions that Adapter for SAP records in its transaction store. Here you can view transactions and their processing log as well as delete transactions that are no longer needed.

Note:

Adapter for SAP does not support XAResource transaction (XA_TRANSACTION).

Viewing Transactions

You view transactions for a local Transaction Store and a Centralized Transaction Store (CTS) or a Shared Transaction Store (STS) in the same way. In a local Transaction Store, you can view only the transactions executed by that individual adapter; in a CTS or STS, you can view the status of all the transactions executed by any adapter in the group.

➤ **To view the transactions in the transaction store**

1. In the Adapters menu in the Integration Server Administrator's navigation area, click **Adapter for SAP**.
2. In Adapter for SAP menu, click **Transactions**. The Transactions screen displays all transactions.
3. By default, Adapter for SAP displays transactions in pages of 20, in order to avoid a timeout or out-of-memory of the web browser in case there is a large number of transactions in the store. To display more or fewer than 20 transactions per page, specify a value in the **Page Entries** field.
4. To filter the displayed list of transactions:
 - a. Enter filter criteria in any of the following fields in the Transaction List Filter section of the Transactions screen: **Sender**, **Receiver**, **Message Type**, **TID**, **State**, **Begin Date**, and **End Date**.

The **Begin Date** and **End Date** fields can be used to restrict the displayed transactions to a specific time interval. The date format entered into these fields must match the date format displayed in the **Date** field in the Transaction List.

When you enter a filter term in one of the fields, only those transactions that match the term will be displayed. Regular expressions are supported to allow flexible filter criteria matching. For a detailed description of regular expressions, see *webMethods Service Development Help* for your release.

- b. Click **Apply Filter**. The transactions that match the filter criteria are displayed.

Note:

Clicking **Apply Filter** runs your filter criteria against all transactions. If you want to refine your search, specify additional or more restrictive filter criteria, and then click **Apply Filter**.

- c. To re-display all transactions, clear all filter fields and click **Apply Filter**.
5. To view detailed information for a transaction:
 - a. For the transaction you want to view, click its corresponding transaction ID in the **TID** column.
 - b. The TID screen displays the transaction ID, sender, receiver, message type, date when this transaction was first received by Adapter for SAP and time when the current state was set, current state, the last error (if any), and an audit log that shows state changes, error messages and the various steps that have processed the message.

The **State** 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 sent a transaction ID, which was accepted by Adapter for SAP transaction manager. (No data sent yet.)	The sender has sent 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. The Administrator should manually follow up transactions in status Rolled back as follows:	Execution of the transaction has failed. The sender may retry the transaction again at a later time.

Status	Meaning (tRFC)	Meaning (other protocols)
	<p>Try to find and eliminate the reason of failure and then:</p> <ul style="list-style-type: none"> ■ 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. 	
Committed	The sender has acknowledged that the transaction executed successfully.	Execution of the transaction on Adapter for SAP 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 <code>pub.sap.transport.*:InboundProcess</code> 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.

- c. If the storing of message bodies has not been disabled on the Routing or RFC listener, it is also possible to view the transaction's complete pipeline or its message body as either XML or HTML (IDoc only) from the View Transaction screen.
- d. To delete this transaction, click **Delete**.
- e. To return to the list of transactions, click **Return to Transaction List**.


Deleting Transactions

You delete transactions from a local Transaction Store and a Centralized Transaction Store (CTS) or a Shared Transaction Store (STS) in the same way. If a transaction store contains transactions that are no longer needed, you can delete them. In a local Transaction Store, you can delete only the transactions executed by that individual adapter; in a CTS or STS, you can delete any transaction executed by any adapter in the group.

Important:

If you delete a transaction in a CTS or STS, the transaction will not be available for all the adapters in the group. Make sure you delete only the transactions that are not needed by any adapters in the group.

➤ **To delete transactions from the transaction store**

1. In the Adapters menu in the Integration Server Administrator's navigation area, click **Adapter for SAP**.
2. In Adapter for SAP menu, click **Transactions**.
3. Use any of the following ways to delete transactions:
 - To delete transactions one at a time, click  in **Delete** column corresponding to the transaction that you want to delete.
 - To delete only the transactions that match your filter criteria, all at the same time, click **Delete Filtered Transactions**.
 - To delete all transactions in the transaction store at once, click **Delete All Transactions**.

Automatic Cleanup of the Transaction Stores

As the size of the transaction store increases, you should consider creating a flow service that periodically purges the store of transactions that are no longer needed.

- To clean up a local Transaction Store, configure the flow service on each adapter.
- To clean up a Centralized Transaction Store or a Shared Transaction Store, configure the flow service on only one of the adapters for SAP in the adapter group. For best performance this should be Adapter for SAP that is the CTS server.

➤ **To purge certain transactions from the transaction store periodically**

1. In Digital Event Services, create a flow service that invokes the `pub.sap.transaction:sweep` service.
2. Pass the following parameters to the sweep service using the **Set Value** modifier in the pipeline.

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 <code>pub.sap.transaction:sweep</code> runs.

For information about the `sweep` service, see [“pub.sap.transaction:sweep” on page 292](#).

3. In the IS Administrator UI go to **Server > Scheduler** and schedule this flow service to run periodically.

Example

To periodically purge all transactions that 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 too much load on Integration Server at one single time. It is also a good idea to schedule the job at times when there is not much load on Integration 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 for a local Transaction Store and for a Centralized Transaction Store Server only. Some of these settings influence the performance of the transaction store.

Note:

The transaction store performance will influence the general performance of Integration Server depending on the transactional throughput, that is, size of the docs and docs per second.

- 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", Adapter for SAP does 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", Adapter for SAP does not maintain any transaction and status information in the transaction store.
- 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", Adapter for SAP links the IDoc packet's TID with the DOCNUMs of the IDocs in that packet, so that later ALE IDoc Monitoring is possible. See [“Using the ALE Monitoring Features Via Adapter for SAP” on page 183](#) for more information.
- `watt.sap.xtn.cacheFlushPeriod`

This property specifies how often Adapter for SAP will flush the unsaved changes in the transaction cache to the file system. The default is 2 seconds; the maximum allowed value is 100 seconds. If a transaction in the cache has been modified (for example: new state or new audit log), it will be persisted to the transaction store after "cacheFlushPeriod" has expired and

no further modifications have happened. After that, the transaction remains in the cache until the "cacheTimeToLive" period has expired, or longer if it was modified again.

Important:

For a Centralized Transaction Store, Adapter for SAP automatically limits the cache flush period to .2 seconds.

■ `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 the amount of time specified in `cacheTimeToLive`. The default is 50 seconds; the maximum allowed value is 500 seconds. If the transaction is not accessed during this time, it is removed from the cache after this period has ended. This keeps the cache from growing indefinitely.

Important:

For a Centralized Transaction Store, Adapter for SAP automatically limits the cache time-to-live period to .2 seconds.

■ `watt.sap.xtn.fastAsyncMode`

This property, when set to "true", enables the synchronous write cache and when set to "false" disables the cache. This property works together with the `watt.sap.xtn.cacheFlushPeriod` and `watt.sap.xtn.cacheTimeToLive`. Setting this feature to "true" will force the transaction manager not to persist each update of a transaction but will keep the state changes in memory and persist it periodically.

Important:

For a Centralized Transaction Store, you cannot set this parameter to "true".

Using the ALE Monitoring Features Via Adapter for SAP

You can monitor the ALE transactions for a local Transaction Store and for a Centralized Transaction Store (CTS) in the same way, as described below. The behavior of monitoring will not change because the transactions are executed by individual adapters even in a CTS setup. Only the transaction results are stored in a CTS server.

When an IDoc is sent from an SAP system to Adapter for SAP, 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 Adapter for SAP and who the final recipient was. To get this kind of information, Adapter for SAP 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 must 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.

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 Integration Server, must contain an outbound parameter with message type SYNCH.
- Also the Function Call to an LS of type "tRFC Port", like 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 Adapter for SAP for IDoc Trace

To set up Adapter for SAP 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 Digital Event Services.
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 Adapter for SAP 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, Adapter for SAP 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) Adapter for SAP 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 Adapter for SAP

➤ To set up Adapter for SAP 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 Digital Event Services.
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:swapSenderReceiverLSas` preprocessing service to the routing notification service.

Status Update Via SYSTAT IDoc

Overview

In this case 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 (Adapter for SAP)
- Routing notification or ALE adapter notification that processed the IDoc
- The key (TID) under which the IDoc can be found in Adapter for SAP
- 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

Adapter for SAP 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"

Set Up the Participating SAP Systems

In each SAP system that is to receive SYSTAT IDocs from 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**
2. **> 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.
3. 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 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 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 Adapter for SAP 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 Adapter for SAP, it creates and submits one SYSTAT IDoc containing all this information.

Shared Transaction Store

As an alternative to storing transaction information locally, Adapter for SAP provides a Shared Transaction Store (STS) configuration that enables you to run several Adapter for SAP in parallel to improve reliability. The STS manages the transaction states for multiple Adapters for SAP, and can be used with either Integration Server cluster or with a group of Adapter for SAP on independent Integration Servers. By providing a central location in which to store transaction information, an STS ensures that the status of incoming tRFCs will remain valid and consistent throughout the adapter group. For more information about using the transaction stores, see [“Considerations about Adapter for SAP Centralized Transaction Store or Shared Transaction Store” on page 55](#)

Adapter for SAP group refers to a number of adapters for SAP that belong logically together, that share a single STS and, by sharing the STS, allow reliable tRFC/IDoc load balancing. There can be only one STS per adapter group. All adapters for SAP that are configured to use a specific STS are considered to be in the same group.

To use the STS, configure all the adapters for SAP to use the same external shared directory as transaction store location. For configuration instructions, see [“Configuring a Shared Transaction Store” on page 191](#).

When a transaction is processed by any of the adapter for SAP in the adapter group, the adapter stores the transaction status in the external shared directory, which can be accessed by all the other adapters for SAP in the same group. Note that the way in which the transactions are processed is

the same as if they were stored locally, but the status of the processed transactions is stored in the STS Server instead of storing the status locally on each adapter in the group.

Use the Transactions screens to view and manage the transactions that the Adapter for SAP records in the STS. For more information, see:

- [“Viewing Transactions” on page 178](#)
- [“Deleting Transactions” on page 180](#)

For additional performance and configuration tasks, see:

- [“Automatic Cleanup of the Transaction Stores” on page 181](#)
- [“Configuration Parameters for the Transaction Manager” on page 182](#)
- [“Using the ALE Monitoring Features Via Adapter for SAP” on page 183](#)
- [“Configuring a Shared Transaction Store” on page 191](#)

Centralized Transaction Store

As an alternative to storing transaction information locally, Adapter for SAP provides a Centralized Transaction Store (CTS) that enables you to run several adapters for SAP in parallel to improve reliability. The CTS manages the transaction states for multiple adapters for SAP, and can be used with either Integration Server cluster or with a group of adapters for SAP on independent Integration Servers. By providing a central location in which to store transaction information, a CTS ensures that the status of incoming tRFCs will remain valid and consistent throughout the cluster or adapter group. For more information about using the transaction stores, see [“Considerations about Adapter for SAP Centralized Transaction Store or Shared Transaction Store” on page 55](#).

Adapter for SAP group refers to a number of adapters for SAP that belong logically together, that share a single CTS and, by sharing the CTS, allow reliable tRFC/IDoc load balancing. There can be only one CTS per adapter group. All adapters for SAP that are configured to share a specific CTS are considered to be in the same group.

To use the Centralized Transaction Store, configure one of the adapters for SAP as the Centralized Transaction Store Server and configure the rest of the adapters for SAP in the group as Centralized Transaction Store Clients. You configure the Remote Server with an alias of "SAPGroupStore" for each Adapter for SAP of your adapter group (and in exactly the same way). For configuration instructions, see [“Configuring a Centralized Transaction Store \(Server\)” on page 191](#).

When a transaction is processed either by the Centralized Transaction Store Server or by any of the Centralized Transaction Store Clients, the adapter stores the transaction status in the server, which can be accessed by the adapter server and all the adapter clients. Note that the way in which the transactions are processed is the same as if they were stored locally, but the status of the processed transactions is stored in the Centralized Transaction Store Server instead of storing the status locally on each adapter in the group.

Use the Transactions screens to view and manage the transactions that Adapter for SAP records in the CTS. For more information, see:

- [“Viewing Transactions” on page 178](#)
- [“Deleting Transactions” on page 180](#)

For additional performance and configuration tasks, see:

- [“Automatic Cleanup of the Transaction Stores” on page 181](#)
- [“Configuration Parameters for the Transaction Manager” on page 182](#)
- [“Using the ALE Monitoring Features Via Adapter for SAP” on page 183](#)
- [“Configuring a Centralized Transaction Store \(Server\)” on page 191](#)
- [“Viewing Centralized Store Configuration and Status” on page 194](#)
- [“Removing the Configuration of a Centralized Transaction Store” on page 195](#)

To migrate the CTS configuration into an STS configuration

Note:

The Central Transaction Store configuration is deprecated and should be changed into a Shared Transaction Store configuration.

1. Create a directory with write access in the file system which can be accessed by all the adapters for SAP in the adapter group. For improved reliability, this directory could be located on an external high-reliability NAS.
2. For all Adapter for SAP in the adapter group, set the configuration switch `watt.sap.xtn.cts.txstore` to the directory created in the above step.

The switch can be set in Integration Server Administrator. Click **Settings > Extended > Edit Extended Settings**.

In the Extended Settings editor, add `watt.sap.xtn.cts.txstore` property to specify the shared directory.

Click **Save Changes**. The property appears in the Extended Settings list.

3. Remove the CTS configuration as described in [“Removing the Configuration of a Centralized Transaction Store” on page 195](#) for each CTS client in the adapter group. Then shut down all the CTS client adapters.
4. Remove the CTS configuration as described in [“Removing the Configuration of a Centralized Transaction Store” on page 195](#) for the CTS server. Then shut down the CTS server adapter.
5. Copy the complete contents of `packages_directory\WmSAP\txStore` directory of the CTS server adapter to the directory created in Step 1.
6. Restart all the adapters for SAP in the adapter group. The adapters will now use the STS located in the directory created in step 1.

Configuring a Shared Transaction Store

Use the following instructions to configure a Shared Transaction Store Server.

Note:

Perform this procedure for each Adapter for SAP in the adapter group. If one of the adapters is configured to CTS, follow the instructions described in [“To migrate the CTS configuration into an STS configuration” on page 190](#).

➤ To configure the Shared Transaction Store

1. Create a directory with write access in the file system which can be accessed by all the adapters for SAP in the adapter group.

For improved reliability, this directory can be located on a high-available NAS

2. Set the configuration switch `watt.sap.xtn.cts.txstore` to the directory in step 1.
3. From Integration Server Administrator, click **Settings > Extended**.

4. Click **Edit Extended Settings**.

In the Extended Settings editor, add or update the `watt.sap.xtn.cts.txstore` property to the location of the directory created in Step 1.

For example, to set `\\tsclient\D\sharedStore` directory as Shared Transaction Store, specify:
`watt.sap.xtn.cts.txstore=\\tsclient\D\sharedStore`

5. Click **Save Changes**.

The property appears in the Extended Settings list.

6. Restart Integration Server.

After Integration Server restart, Adapter for SAP will use the specified external directory as STS.

If the specified directory does not exist or if it is not writable, an error message will be logged during adapter startup and Adapter for SAP will use the default Local Transaction Store instead.

To disable STS configuration and switch back to Local Transaction Store, remove the `watt.sap.xtn.cts.txstore` property in Extended Settings and restart Integration Server.

Configuring a Centralized Transaction Store (Server)

Use the following instructions to configure a Centralized Transaction Store Server.

Note:

Perform this procedure for Adapter for SAP that is to be the CTS Server.

➤ **To configure the Centralized Transaction Store (Server)**

1. In the Settings menu in the Integration Server Administrator's navigation area, click **Remote Servers**.
2. In the **Remote Servers** page, click **Create Remote Server Alias**.
3. In the **Remote Server Alias Properties**, specify values for the following fields:

Field	Description
Alias	<p>This is the name by which Adapter for SAP will identify the Centralized Transaction Store configuration.</p> <p>You must enter SAPGroupStore in this field.</p>
Host Name or IP Address	<p>This is used to identify Integration Server on which the Centralized Transaction Store is hosted.</p> <p>You must enter the host name or IP address of Integration Server hosting the Centralized Transaction Store, or localhost or 127.0.0.1 if it is the local Integration Server.</p>
Port Number	Enter the port number on which Integration Server is running.
User Name	Enter the user name to log on to Integration Server on which the Centralized Transaction Store is hosted.
Password	Enter the password to log on to Integration Server on which the Centralized Transaction Store is hosted.
Execute ACL	<p>Select the required ACL type from the drop-down list.</p> <p>Default: Internal</p> <p>For more information about assigning and managing ACLs, see <i>webMethods Service Development Help</i> for your release.</p>
Max Keep Alive Connections	<p>Specify the maximum number of repository connections to Adapter for SAP that should be retained. In other words, this is not a maximum number of connections that can be established, but a limit on the number of inactive connections to retain for reuse. If not specified, five keep alive connections are retained.</p> <p>Default: 5</p>
Keep Alive Timeout (minutes)	Delay time until an unused repository connection to Adapter for SAP is timed out.

Field	Description
	Default: 1
Use SSL	Select if you want to use the SSL protocol. Default: No
Retry Server	This field will not be used by the Centralized Transaction Store.

4. Click **Save Changes**.
5. Reload the WmSAP package.
6. Restart Integration Server.

To view the configuration and the status of the store, see [“Viewing Centralized Store Configuration and Status” on page 194](#).

Configuring a Centralized Transaction Store (Client)

Use the following instructions to configure Centralized Transaction Store Clients.

Note:

Perform this procedure for each Adapter for SAP that is to be a CTS client.

➤ To configure the Centralized Transaction Store (Client)

1. In the Settings menu in the Integration Server Administrator navigation area, click **Remote Servers**.
2. In the **Remote Servers** page, click **Create Remote Server Alias**.
3. In the **Remote Server Alias Properties**, specify values for the following fields:

Field	Description
Alias	This is the name by which Adapter for SAP will identify the Centralized Transaction Store configuration. You must enter SAPGroupStore in this field.
Host Name or IP Address	This is used to identify Integration Server on which the Centralized Transaction Store Server is hosted. You must enter the server IP address or the host name of the Centralized Transaction Store.

Field	Description
Port Number	Enter the Centralized Transaction Store Server port number.
User Name	Enter the user name to log on to Integration Server on which the Centralized Transaction Store Server is hosted.
Password	Enter the password to log on to Integration Server on which the Centralized Transaction Store Server is hosted.
Execute ACL	<p>Select the required ACL type from the drop-down list.</p> <p>Default: Internal</p> <p>For general information about assigning and managing ACLs, see <i>webMethods Service Development Help</i> for your release.</p>
Max Keep Alive Connections	<p>Specify the maximum number of repository connections to Adapter for SAP that should be retained. In other words, this is not a maximum number of connections that can be established, but a limit on the number of inactive connections to retain for reuse. If not specified, five keep alive connections are retained.</p> <p>Default: 5</p>
Keep Alive Timeout (minutes)	<p>Delay time until an unused repository connection to Adapter for SAP is timed out.</p> <p>Default: 1</p>
Use SSL	<p>Select if you want to use the SSL protocol.</p> <p>Default: No</p>
Retry Server	This field will not be used by the Centralized Transaction Store.

4. Click **Save Changes**.
5. Reload the WmSAP package.
6. Restart Integration Server.

To view the configuration and the status of the store, see [“Viewing Centralized Store Configuration and Status” on page 194](#).

Viewing Centralized Store Configuration and Status

➤ To view the centralized store configuration and status

1. In the Adapters menu in the navigation area of the Integration Server Administrator, click **Adapter for SAP**.
2. In Adapter for SAP menu in the navigation area of Adapter for SAP, click **Settings**.

At the bottom of the page, you can view the configuration details of **Adapter for SAP Group**. The **Centralized Store** field must display **Local File System** for Centralized Transaction Store Server and **Remote File system <server IP Address>** for Centralized Transaction Store Clients.

Important:

Ensure that Integration Server that hosts the Centralized Transaction Store is always started first, before starting any other Adapter for SAP in Adapter for SAP group. If the Centralized Transaction Store Clients cannot find or access the Centralized Transaction Store Server during adapter startup, they will not be able to start and will shut down with failure. In the same way, it is also important to always shut down all Centralized Transaction Store Clients before shutting down the Centralized Transaction Store Server. The Centralized Transaction Store Clients will otherwise display runtime errors while trying to persist transaction status information, and transactions or transaction changes will be lost.

Removing the Configuration of a Centralized Transaction Store

If you no longer want a particular Adapter for SAP to use the Centralized Transaction Store (CTS), you can remove or rename the adapter's Remote Server with alias **SAPGroupStore** by following the instructions in this section.

If you remove the Remote server for Adapter for SAP that is a CTS Client, the CTS Server will display that particular client's status as **Inactive** in the adapter's **Settings** page. After reloading the WmSAP package or restarting the Integration Server, Adapter for SAP, the adapter will use its local Transaction Store to persist transaction information.


Important:

It is recommended that you do not remove the CTS Server. However, if you do remove the alias for Adapter for SAP that is the CTS Server, none of the CTS Clients will be able to process any transactions and a message is displayed on the console indicating that the CTS Server is not accessible. When Adapter for SAP that is configured as a CTS client cannot connect to the CTS Server during startup, an error message is displayed, the adapter will only be partially loaded, and transactions are not persisted.

Do not remove or rename the Remote Server Alias for **SAPGroupStore** while Adapter for SAP is still processing messages. Otherwise the transaction status of these messages could be lost or become invalid.

➤ From Adapter for SAP that you want to remove from the CTS

1. In the Settings menu in the Integration Server Administrator navigation area, click **Remote Servers**.

2. On the Remote Servers screen, click  for the Remote Server Alias of Adapter for SAP that you no longer want to use the CTS.

Adapter for SAP removes the Remote Server Alias.

3. Reload the WmSAP Package.

The affected Adapter for SAP does not register itself with the CTS server during adapter start-up, and will now use its local Transaction Store instead of the CTS to persist all transactions. By doing so, the adapter is effectively no longer part of the logical group of adapters.

12 Coding Client Applications and Services

■ Overview	198
■ Invoking RFCs from Adapter for SAP	198
■ Receiving IDocs from an SAP System	199
■ Constructing an IDoc with the SAP Java IDoc Class Library	200
■ Transaction Features for HTTP and SAP-XML	202
■ Calling a BAPI Synchronously from SAP System	203
■ Calling a BAPI Asynchronously from an SAP System	206

Overview

Adapter for SAP 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 [“Built-in Services” on page 259](#).

For more information about building the Java services, see the online API documentation installed with Integration Server at:

Integration Server_directory\instances*instance_name*\doc\api\Java\index.html

This chapter also shows how to use webMethods Adapter for SAP IDoc Java API to construct an IDoc. For information, refer to *packages_directory*\WmSAP\pub\doc\api\index.html.

Invoking RFCs from Adapter for SAP

Calling Public Adapter for SAP 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 *webMethods Service Development Help* for your release and the online API documentation.

➤ To call a public Adapter for SAP service from Java

1. Start Designer if it is not already running.

Note:

The steps for using Designer to call the adapter service from Java are similar.

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();
}
```

3. 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 Adapter for SAP 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 145](#). 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 120](#).

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();
com.sap.conn.idoc.IDocDocumentList iDocList=null;
if (idc.first("iDocList"))
    iDocList=(com.sap.conn.idoc.IDocDocumentList) idc.getValue();
```

under **Shared > Imports**, include the lines:

```
com.wm.adapter.sap.idoc.*
com.sap.conn.idoc.*
```

Note:

To resolve the included Java classes, you must include the WmSAPpackage 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 200](#).

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.conn.idoc.*` in **Shared > Imports**.

```
public final static void createIDoc (IData pipeline)
throws ServiceException
{
//create a new and empty MATMAS02 document
try
{
com.sap.conn.idoc.IDocDocumentList iDocList = new IDataDocumentList("MATMAS02");
IDocDocument doc = iDocList.addNew();
//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.setValue("MSGFN", "005");
segment.setValue("MATNR", "BOXCOOKIES");
segment.setValue("ERSDA", "20030303");
segment.setValue("ERNAM", "TOPSI");
segment.setValue("PSTAT", "KBG");
segment.setValue("MTART", "FERT");
segment.setValue("MBRSH", "L");
segment.setValue("MATKL", "G1113");
segment.setValue("MEINS", "PCE");
segment.setValue("BLANZ", "000");
segment.setValue("BRGEW", "0.550");
segment.setValue("NTGEW", "0.000");
segment.setValue("GEWEI", "KGM");
segment.setValue("VPSTA", "KBG");
//create and add a new and empty child segment of type E1MAKTM
//and fill the segment data
segment = segment.addChild("E1MAKTM");
segment.setValue("MSGFN", "005");
segment.setValue("SPRAS", "D");
segment.setValue("MAKTX", "Schachtel mit Keksen");
segment.setValue("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.setValue("MSGFN", "005");
segment.setValue("SPRAS", "E");
segment.setValue("MAKTX", "Box of cookies");
segment.setValue("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.setValue("MSGFN", "005");
segment.setValue("WERKS", "0001");
segment.setValue("PSTAT", "BG");
segment.setValue("PLIFZ", "0");
segment.setValue("WEBAZ", "0");
segment.setValue("PERKZ", "M");
segment.setValue("AUSSS", "0.00");
segment.setValue("BESKZ", "E");
segment.setValue("AUTRU", "X");
//create and add a new and empty sibling segment of type E1MBEWM
//and fill the segment data
segment = segment.addSibling("E1MBEWM");
segment.setValue("MSGFN", "005");
segment.setValue("BWKEY", "0001");
segment.setValue("VPRSV", "S");
segment.setValue("VERPR", "0.00");
segment.setValue("STPRS", "15.50");
segment.setValue("PEINH", "1");
segment.setValue("BKLAS", "7920");
segment.setValue("VJVPR", "S");
segment.setValue("VJVER", "0.00");
segment.setValue("VJSTP", "15.50");
segment.setValue("LFGJA", "2002");
segment.setValue("LFMON", "08");
segment.setValue("PSTAT", "BG");
segment.setValue("KALN1", "000100126602");
segment.setValue("KALNR", "000100126603");
segment.setValue("EKALR", "X");
segment.setValue("VPLPR", "0.00");
segment.setValue("VJBKL", "7920");
segment.setValue("VJPEI", "1");
segment.setValue("BWPEI", "0");
IDataCursor idc=pipeline.getCursor();
IDataUtil.put(idc, "iDocList", iDocList);
idc.destroy();
}
catch (Exception e)
{
    throw new ServiceException(e);
}

```

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:createTIDservice`.

2. Send the IDoc to the SAP system by invoking the `pub.sap.client:sendIDoc` service, passing in the `com.sap.conn.idoc.IDocDocumentList` 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 Adapter for SAP, 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>
```

```
</sap:Body>
</sap:Envelope>
```

Calling a BAPI Synchronously from 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 Adapter for SAP as described in [“Listeners” on page 94](#).
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 91](#).
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 Adapter for SAP instructions for routing
data header like SBCCALENV occurs 1 with header line.
*variable returnCode takes the processing information after the
*synchronous call
data returnCode type BAPIRETURN.
*Fill the Adapter for SAP 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 Adapter for SAP
  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.
```

Preparing a Routing Notification for the BAPI Call

Before executing the report in the section [“Calling a BAPI Synchronously from SAP System” on page 203](#), 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 Designer and select **New > Adapter Notification**. Click **Next**.
2. Select **Adapter for SAP** 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.

Specifying a Second Routing Notification for the BAPI

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, Adapter for SAP 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: ssid=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>0A125F1315B3D24B00000001E</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 Adapter for SAP. 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 Adapter for SAP 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 address 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.

```

Creating Similar Routing Notifications

➤ 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:
 - **Sender:** enter the logical system of the sending system.
 - **Receiver:** enter the logical system of the target system.
 - **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.

Handling 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.

➤ To handle the Inbound XML Message

1. In the input field, enter the following data:
 - **Sender:** enter the logical system of the sending system.
 - **Receiver:** enter the logical system of the target system.
 - **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 Adapter for SAP.

Example of an XML

When executing the report in the SAP system, Adapter for SAP 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>0A125F1315B3A11B000000035</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="">
```

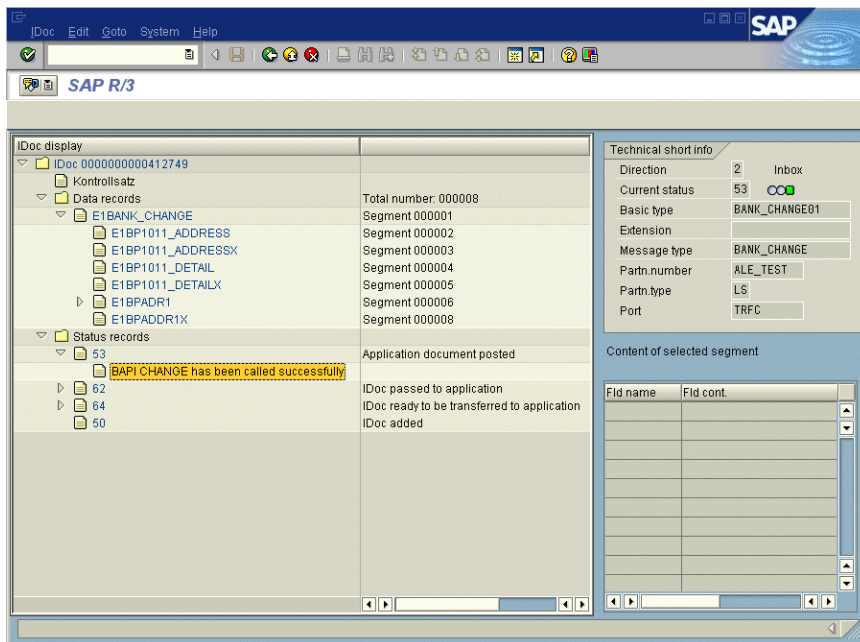


```

<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.

13 Security

■ Adapter for SAP Configuration	212
■ User Authentication Between Adapter for SAP and an SAP System	212
■ Installing Adapter for SAP According to Your Security Policy	217

Adapter for SAP Configuration

For information on how to configure your Integration Server securely, please refer to the *webMethods Integration Server Administrator's Guide* for your release.

User Authentication Between Adapter for SAP 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.

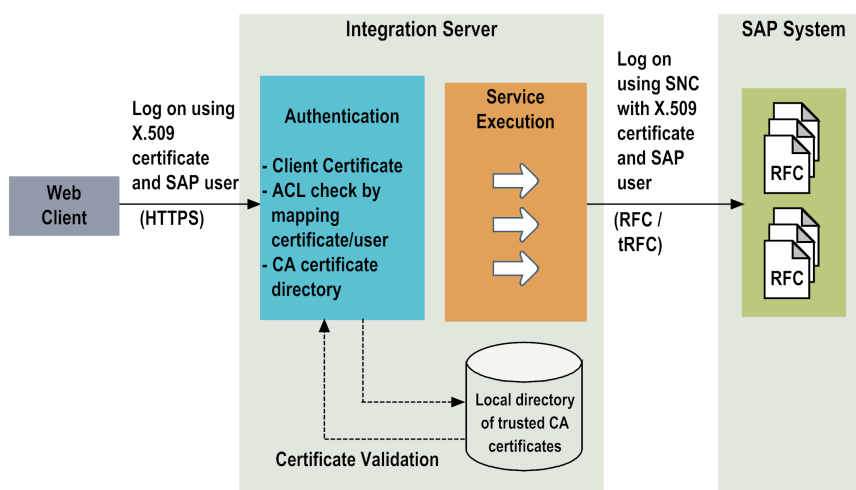
Authentication Through X.509 Certificate

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 Adapter for SAP with the SAP Cryptographic Library for SNC” on page 216](#) for more information about SNC and adapters for SAP.

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 Adapter for SAP and on the SAP system. These settings are listed in the section [“Configuring Adapter Connections” on page 58](#). For detailed information on the SAP system and RFC client settings for SNC see the corresponding SAP documentation.

Important:

The HTTPS port defined on Integration Server should have the **Request Client Certificate** option set for the **Client Authentication** field.

Important:

SNC connections opened with the X.509 certificate are locked to the HTTPS session and will remain open until the HTTPS session is closed.

Tip:

For more information on ports, see *webMethods Integration Server Administrator's Guide* for your release.

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 Integration Server.

Important:

Before you can log on to Integration Server using a trusted certificate, you have to import the (personalized) client certificate for each user from a local directory to Integration Server and map it to 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 *Integration Server_directory* directory.

When a user logs on (for example, from a Web client) using this certificate, 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 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 *webMethods Integration Server Administrator's Guide* for your release.

Example

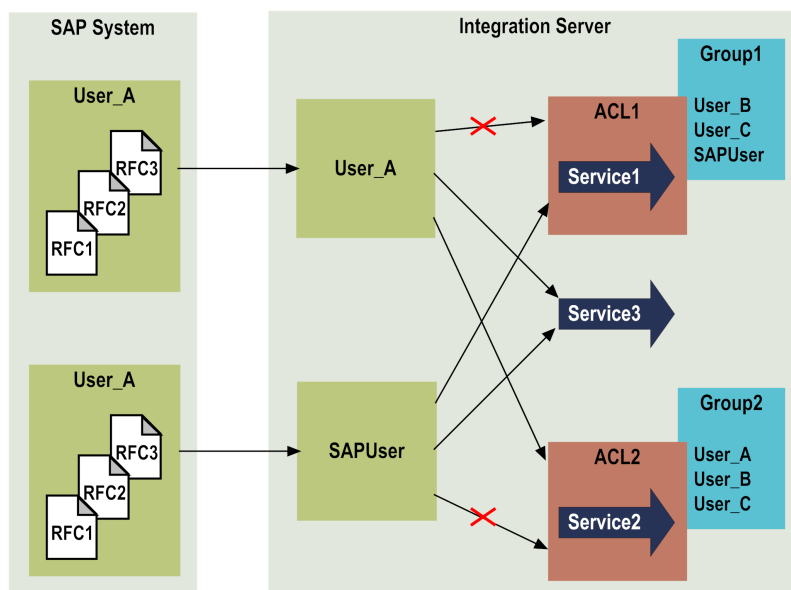
You want to execute a service on Integration Server that retrieves sales order data from an SAP system. This service is protected by an ACL. Integration Server user 'Sales' is registered in this ACL and allowed to execute the service. If you map the certificate to the user 'Sales', your SAP user can 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 Adapter for SAP 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 Adapter for SAP Acts As an RFC-Server

Mapping SAP User to Integration Server User



If Adapter for SAP is called from an SAP system, the call is always trusted. Therefore, only the user name from the SAP system is used for the login. 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 94](#) 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 Adapter for SAP 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 Integration Server and Adapter for SAP 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 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).

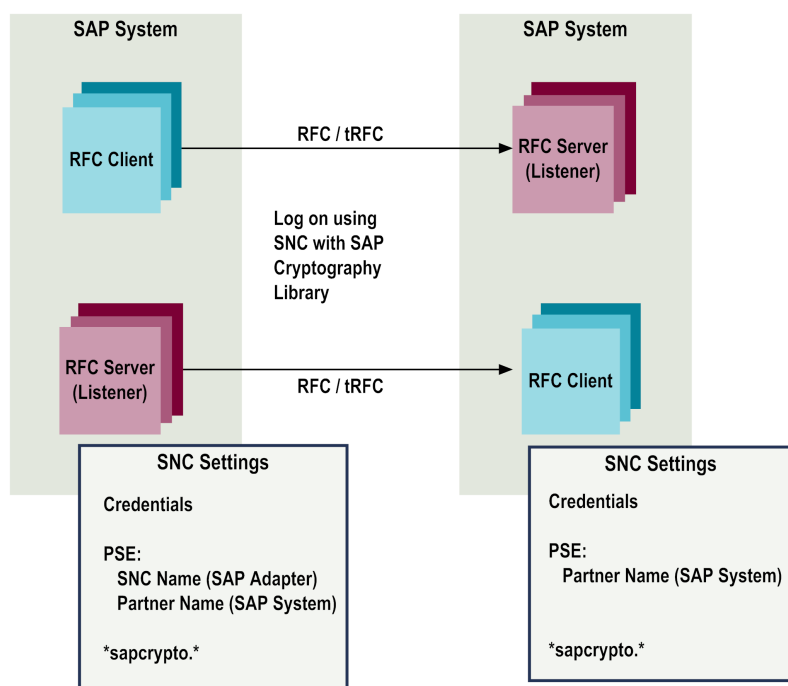
Important:

The user names in Integration Server are case sensitive.

Using Adapter for SAP 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 Adapter for SAP communication using the `sapcrypto.lib`. For more detailed information on the installation of `thesapcrypto.lib` and the generation of PSE (Personal Security Environment) see *Using the SAP Cryptographic Library for SNC* in the SAP Online Documentation (SAP Library).

Adapter for SAP Communication Using the SAP Cryptographic Library for SNC



➤ To use Adapter for SAP 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 Adapter for SAP and the corresponding SAP system for SNC use (see Security Options in the section [“Configuring Adapter Connections” on page 58](#) and [“Listeners” on page 94](#)).
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 Adapter for SAP According to Your Security Policy

Adapter for SAP 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 Integration Server and the SAP systems or putting Integration Server in the DMZ. You can configure the firewall to restrict which SAP systems can be accessed from Adapter for SAP through the SAP router.

Finally, you might even want to completely disallow Integration Server in the DMZ to actively open connections to a SAP system in the intranet. To do so, you need to install two Integration Servers: one in the DMZ, which is configured as an Enterprise Gateway Server, and one in the intranet. Integration Server in the intranet establishes the connection to Enterprise Gateway Server, whereas data still flows synchronously from the outside to the inside. For information about how to configure Integration Server as an Enterprise Gateway Server, see *webMethods Integration Server Administrator's Guide* for your release.

14 Managing the DDIC Cache

■ Data DICTIONary Cache (DDIC Cache)	220
■ Viewing Information in the DDIC Cache	220
■ Removing Information from the DDIC Cache	222

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. Adapter for SAP retrieves this information from an SAP system when it performs RFC, BAPI, or IDoc lookups. To improve performance, Adapter for SAP caches information it receives about function modules and structure definitions.

Adapter for SAP receives information about function modules and structure functions when:

- An RFC, BAPI, or IDoc lookup is requested from the **Lookup** screen.
- Adapter for SAP 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 Adapter for SAP.

Adapter for SAP keeps separate cached function modules, structure definitions, Business Objects, ALE mappings, and IDocs for each SAP system. When Adapter for SAP 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, Adapter for SAP 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 Integration Server.

Note:

The DDIC cache does not persist through shutdown and restart of 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 **Adapter for SAP**.
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 58](#).

Note: If you have configured and enabled an RFC listener, you will also see a 'Local' entry in the **System ID** list. The cached elements of this **System ID** cannot be deleted or modified.

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.

Adapter for SAP 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.

Adapter for SAP 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 Adapter

for SAP tries only once to retrieve ALE information. If this retrieval fails, Adapter for SAP retains this fact.

8. After enabling one (or more) RFC listeners, a new entry named LOCAL is listed in the **DDIC Cache** screen of Adapter for SAP. The LOCAL entry contains four function definitions: **WM_PUSH_START_EVENT**, **WM_PUSH_STOP_EVENT**, **WM_PUSH_ERROR_EVENT**, and **WM_PUSH_BUSINESS_DATA** and two structure definitions: **WMKEYVALUE** and **WMBUSINESSDATA**.

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 **Adapter for SAP**.
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 ✖ in **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 **Adapter for SAP**.
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.

Note:

The default entries of the LOCAL entry cannot be deleted.

3. Click **Return to DDIC-Cache** to return to the SAP Repository Cache table.

15 Managing SAP User Store

■ Managing the SAP User Store	226
■ Adding Entries to SAP User Store	228
■ Changing Entries in SAP User Store	230
■ Removing Entries from SAP User Store	232

Managing the SAP User Store

Note:

Adapter for SAP application typically executes different RFC client calls under different SAP user accounts. If the RFC client call needs to be executed with a different SAP user than the default user, it is recommended to create one RFC connection for each connected SAP system, and to override the SAP username and password during runtime.

Adapter for SAP version 7.1 provides the *\$user* and *\$pass* input parameters for all RFC Adapter Services and for the public client services "pub.sap.client:*" to implement overriding the default user at runtime.

With the previous Adapter for SAP versions, you have to store the SAP user names and SAP passwords as cleartext in *\$user* and *\$pass* input parameters in the application services.

Adapter for SAP 10.1 extends the user overriding functionality by providing an indirect SAP **User Alias** mechanism for the RFC client call execution. It provides an SAP User Store user interface to maintain the SAP user aliases together with their associated SAP user names and passwords.

The indirect SAP **User Alias** mechanism allows you to :

- Remove the SAP passwords from the application services
- Change the SAP username and password centrally without modifying the application services.

Note:

An SAP **User Alias** is a unique key name for an SAP user. The SAP User Store holds a list of SAP **User Alias** entries. Each **User Alias** entry is associated to an SAP username and password.

The **User Alias** name is used during runtime to retrieve the SAP username and password from the SAP User Store.

The SAP User Store user interface allows to add, modify and delete the required SAP **User Alias** entries.

Runtime Behavior

Note:

The extended overriding functionality in Adapter for SAP 10.1 uses the existing (optional) *\$user* and *\$pass* input parameters so that there are no changes in the service signatures required.

If both the *\$user* and *\$pass* values exist in the input pipeline, the runtime behavior remains unchanged. Adapter for SAP 10.1 will interpret the values as cleartext username and password in the same way as previous versions of Adapter for SAP. Existing Adapter for SAP application will show the same runtime behavior with Adapter for SAP in this case.

However, if only *\$user* value exists in the pipeline and *\$pass* is undefined, the Adapter for SAP 10.1 will take the *\$user* value as the **User Alias** name to retrieve the SAP username and password from the SAP User Store.

Note:

To use the extended overriding functionality for RFC client calls, the *\$pass* value must be removed from the input pipeline, and the *\$user* value must be changed to a valid SAP **User Alias** name.

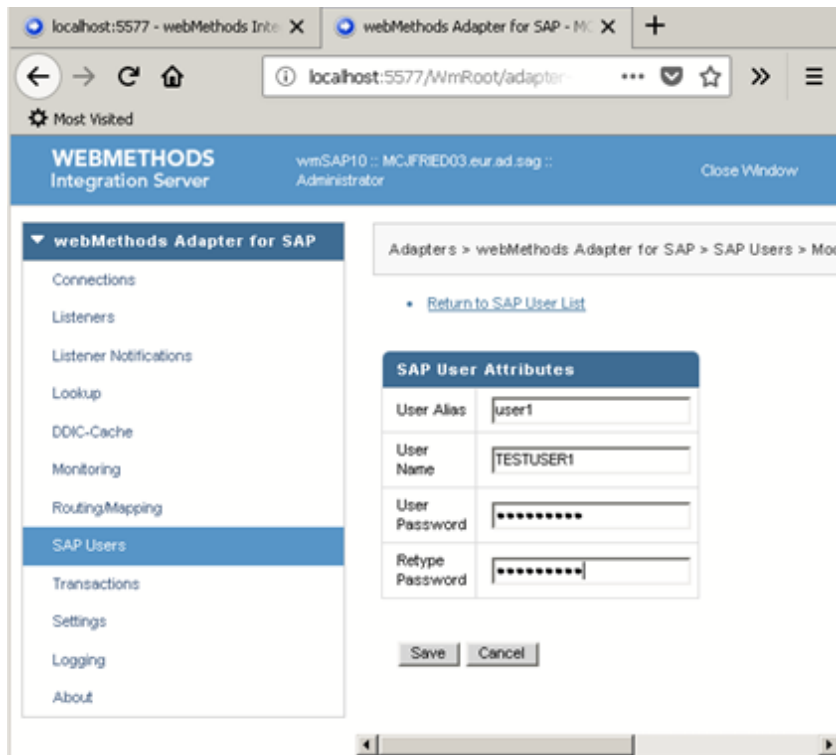
Differences in the runtime behavior when executing RFC client services

Pipeline	Adapter for SAP 7.1	Adapter for SAP 10.1
Empty. No values for <i>\$user</i> and <i>\$pass</i>	Service will be executed with the default user of the RFC connection.	Service will be executed with the default user of the RFC connection.
<i>\$user</i> and <i>\$pass</i> exist in the pipeline	Service will be executed with the provided user if the SAP user name exists and the password is valid.	Service will be executed with the provided user if the SAP user name exists and the password is valid.
<i>\$user</i> and <i>\$pass</i> exist in the pipeline, but the logon information is incorrect.	Service execution will cause an error with the error message JCO_ERROR_LOGON_FAILURE "Name or password is incorrect" or "Incomplete logon data" if the password is missing.	Service execution will cause an error with the error message JCO_ERROR_LOGON_FAILURE, "Name or password is incorrect".
<i>\$user</i> exists in the pipeline, but <i>\$pass</i> does not exist.	Service execution will cause an error with the error message JCO_ERROR_LOGON_FAILURE "Incomplete logon data".	<p>Adapter for SAP 10.1 will use <i>\$user</i> as User Alias name to look-up the SAP user name and password in the User Store:</p> <ul style="list-style-type: none"> ■ If the alias does not exist, there will be an error message in the IS log "SAP user alias "XXX" does not exist!" and the service execution will fail with a JCO_ERROR_LOGON_FAILURE error. ■ If the alias exists in the User Store and the associated SAP user name and password are valid, then the service will be executed with this SAP user. 3. If the alias exists but the associated user information is not valid then the service will fail with JCO_ERROR_LOGON_FAILURE error.

Adding Entries to SAP User Store

After the installation of Adapter for SAP, the SAP User Store is initially empty and the administration user has to add the required SAP user names and passwords and provide each entry with a unique SAP **User Alias** name.

1. In Integration Server Administrator Adapters menu, click Adapter for SAP.
2. Click **SAP Users**, then click **Add New User Link**.

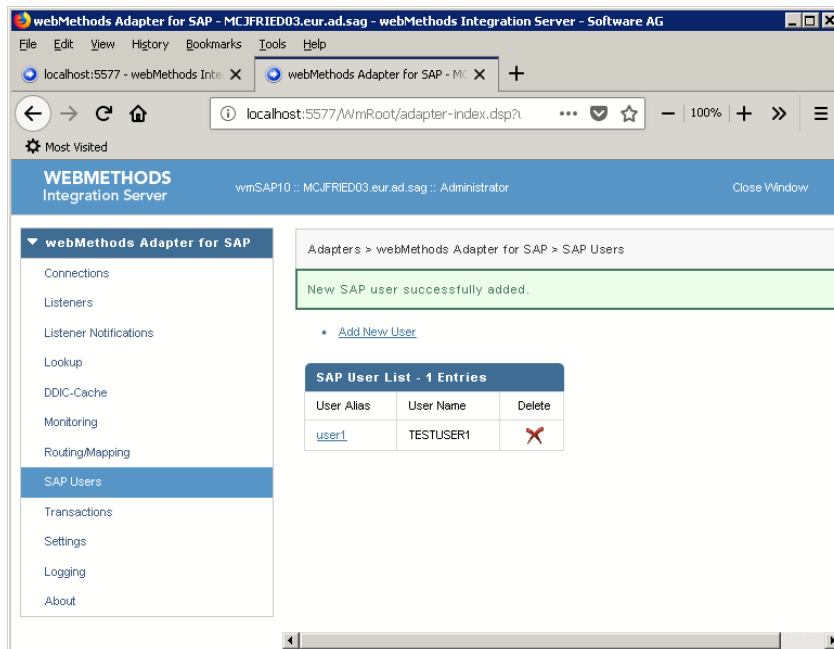


The example screen above shows the creation of a new SAP User Store entry with the new SAP **User Alias** name being 'user1', the SAP **User Name** being 'TESTUSER1' and the SAP **User Password** which is not displayed in clear text.

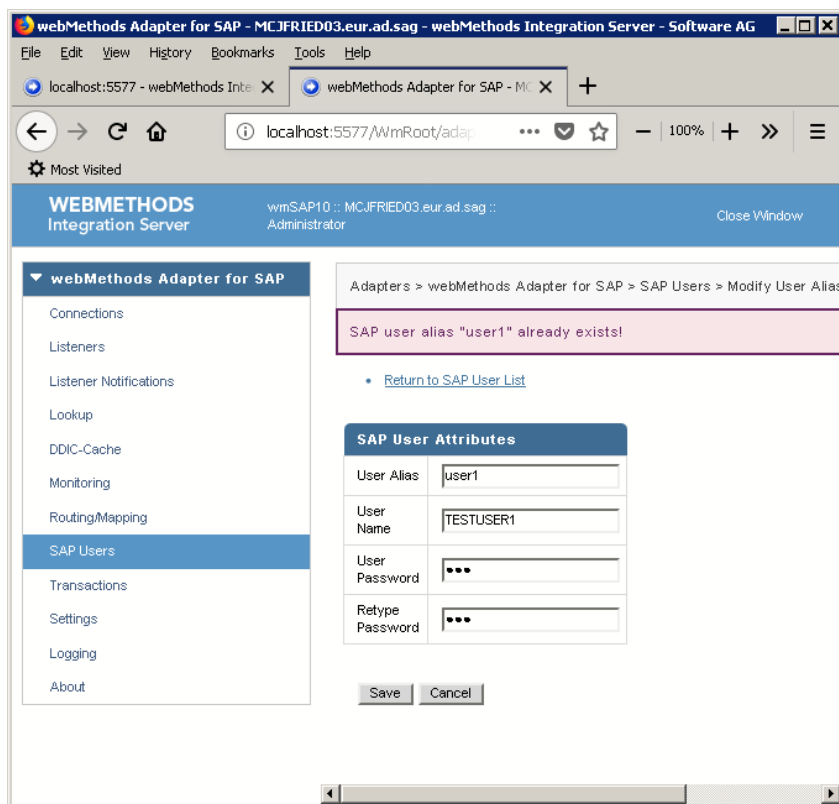
Clicking on **Cancel** discards the new entry and returns to the main SAP User Store UI screen.

Clicking on **Save** lets Adapter for SAP check if the entries are valid and that the **User Alias** name has not been defined before.

If the SAP **User Name** and **User Password** entries are valid (not empty) and the **User Alias** name is new and unique then the new entry will be added to the SAP User Store and will be displayed in the SAP User Store main UI screen:



If the new **User Alias** name already exists in the User Store or if the SAP **User Name** or SAP **User Password** values are empty then the new entry is not added to the User Store and an error message is displayed. Failures with error messages occur in case of unequal password entries, in case of missing password entries and if the new SAP User Alias already exists in the User Store:



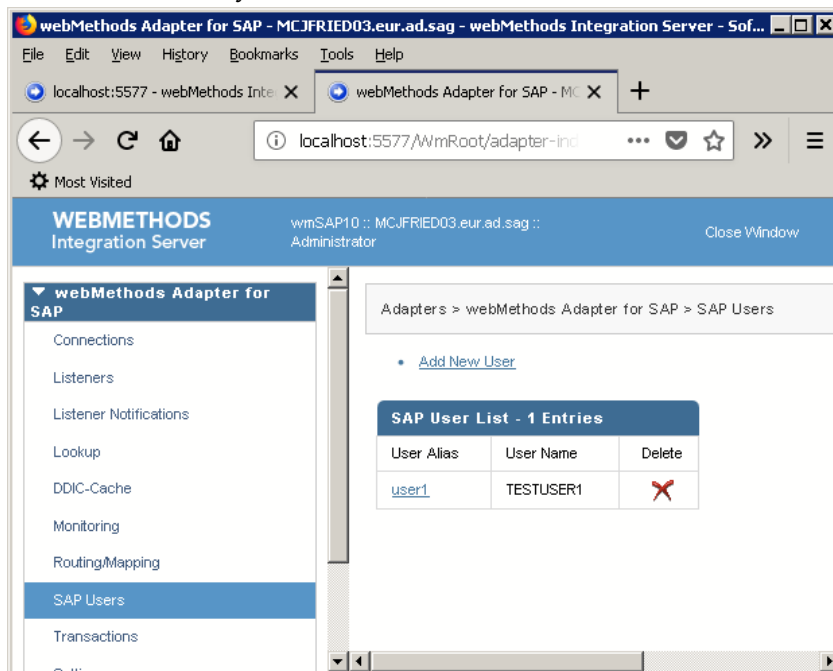
Note:

Only the **User Alias** name must be unique, but the SAP **User Names** may occur multiple times in the User Store. It is possible to create several different **User Alias** names which are associated to the same SAP user name and password.

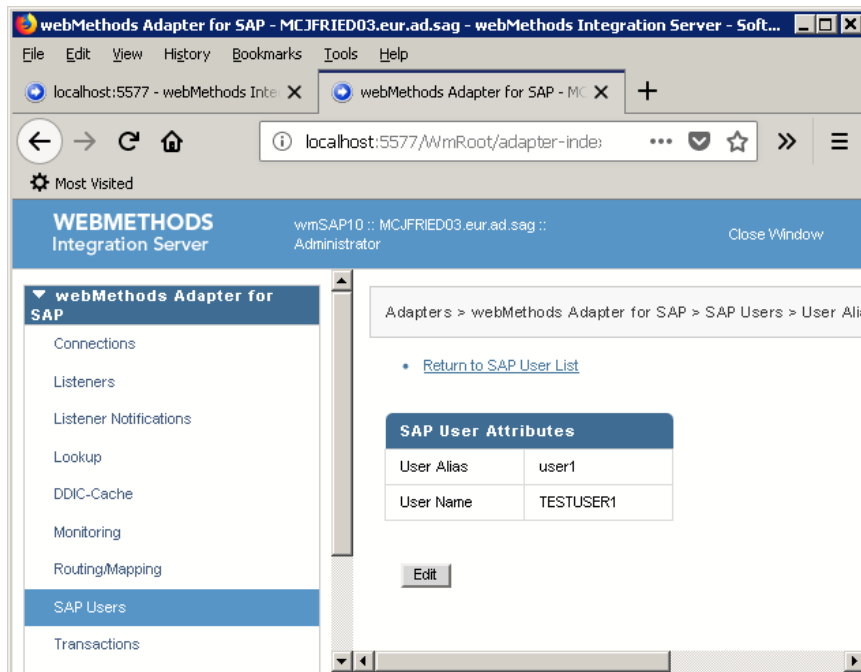
Changing Entries in SAP User Store

1. In Integration Server Administrator Adapters menu, click Adapter for SAP.
2. Click **SAP Users**, then click **SAPUser Alias**.

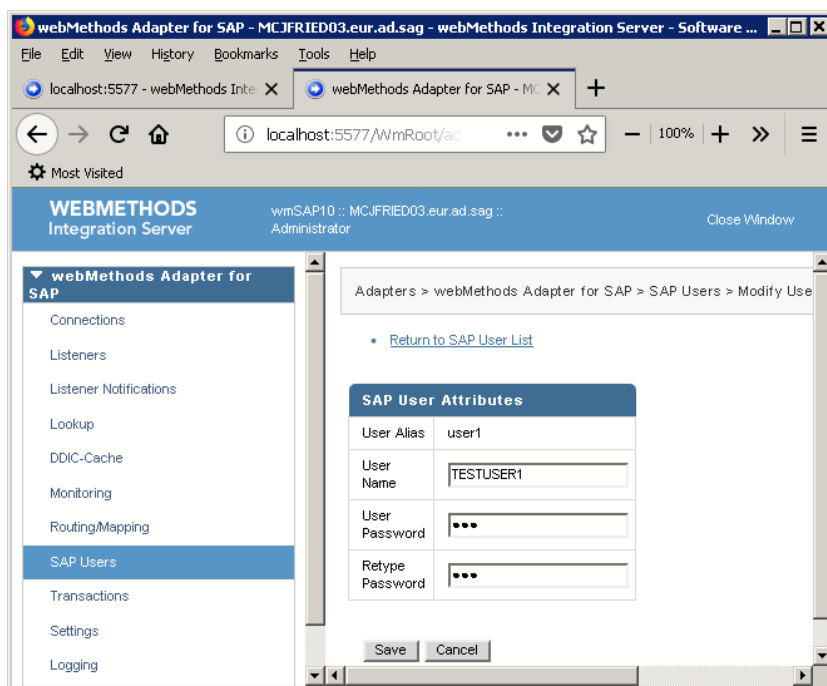
In User Store entry table select the user to be modified. For example, user1.



This leads to a screen displaying the user attributes of the entry.



3. Click **Edit** to change the **User Name** and **User Password** values.



4. You can either **Cancel** or **Save** the changes by:
 - Click **Cancel** to cancel all changes.
 - Click **Save** to let Adapter for SAP verify the changed values and persist them in the User Store, if they are valid.

Note:


When changing the existing password, the new password must be entered twice.

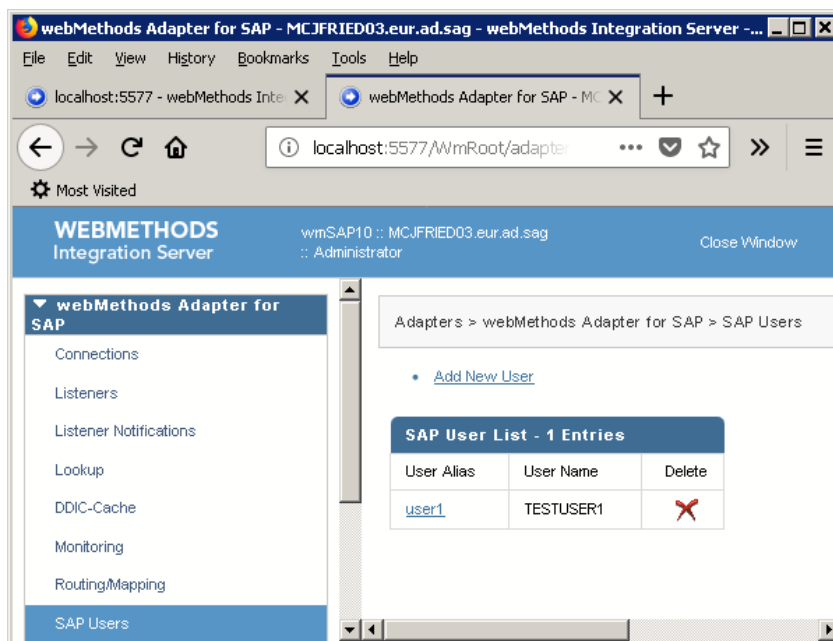
It is not possible to change the **User Alias** in the **Edit** screen. In order to change the User Alias the existing Alias has to be deleted and then a new User Alias entry has to be created with a new **User Alias name**.

Note:

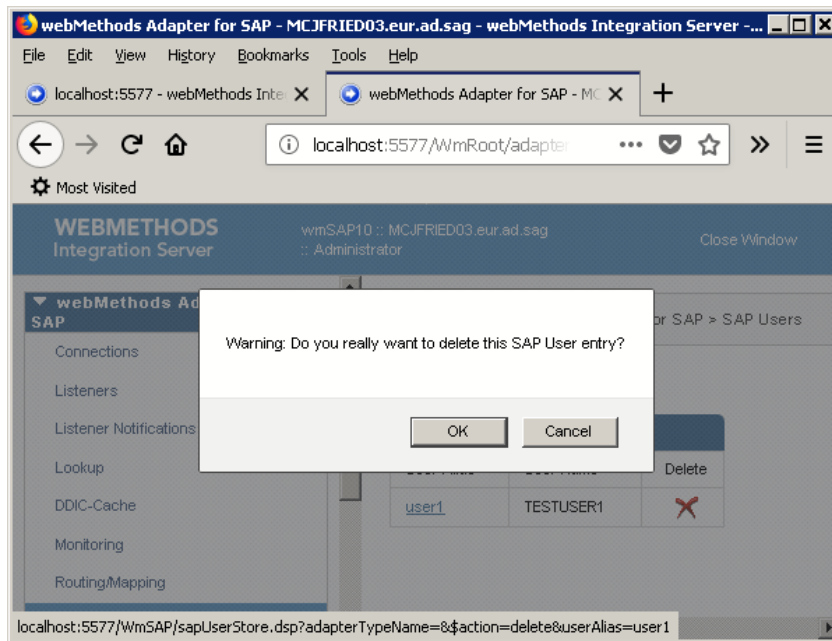
Adapter for SAP does not check if new or updated SAP user names and passwords are valid and if they exist in any SAP system. The User Store does not depend on specific SAP Systems and it therefore cannot check the validity of the user entries. Whether a SAP user name or password is valid can only be determined during runtime.

Removing Entries from SAP User Store

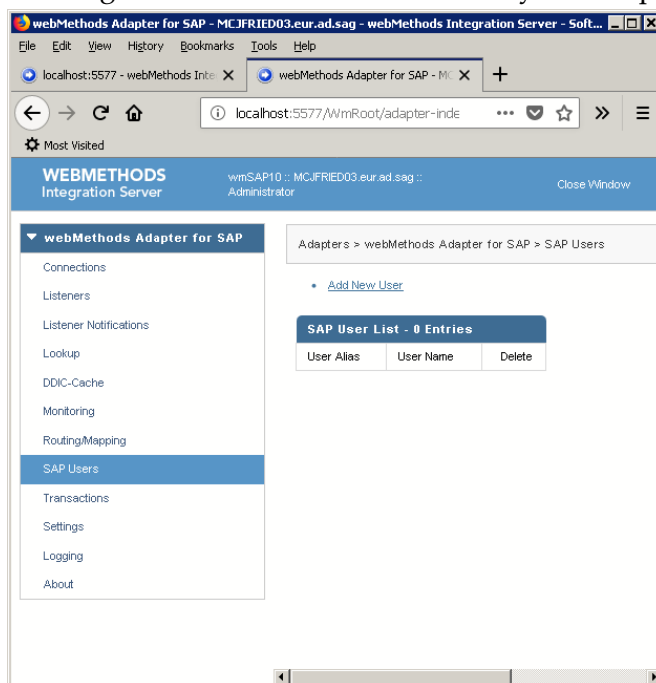
1. In Integration Server Administrators menu, click Adapter for SAP.
2. Click **SAP Users**, then click  for the **User Alias** entry to be removed.



3. Confirm the deletion by clicking **OK**. Click **Cancel** to cancel the deletion.



Clicking on **OK** deletes the selected entry and displays the updated content of the User Store.



16 Predefined Health Indicator

■	Predefined Health Indicator	236
---	-----------------------------------	-----

Predefined Health Indicator

Microservices Runtime includes predefined health indicators for some of its basic components. The health indicator captures the connection details for all the WmART based adapters at runtime. For more information, see *webMethods Adapter Runtime User's Guide*.

17 Administrator APIs

■ Administrator APIs	238
----------------------------	-----

Administrator APIs

The Administrator APIs are available for Adapter for SAP. For more information about Administrator APIs and samples, see *webMethods Adapter Runtime User's Guide*.

18 Configuration Variables Templates for Adapter Assets in Microservices Runtime

■	Configuration Variables Templates for Adapter Assets in Microservices Runtime	240
---	---	-----

Configuration Variables Templates for Adapter Assets in Microservices Runtime

The webMethods Adapter Runtime (ART) asset properties that can be configured from Integration Server Administrator are available in the configuration variables template (`application.properties` file) generated by Microservices Runtime. For more information, see *webMethods Adapter Runtime User's Guide* and *Developing Microservices with webMethods Microservices Runtime*.

19 Logging and Monitoring

■ Overview	242
■ Logging	242
■ Monitoring Statistics for Each Adapter for SAP Connection	243
■ Monitoring Adapter for SAP Performance	245

Overview

The following sections describe logging information and monitoring Adapter for SAP connection statistics and performance.

Logging

Adapter for SAP produces detailed logging information using 19 different log facilities. The log level of Log facilities can be individually disabled or set to different log level, and the log level can be set from "Fatal" (logs only critical information) to "Trace" (logs verbose information) for the enabled log facilities. The log level determines the amount of logging information being produced by the adapter. Please note that a high log level will reduce the performance of the adapter's processing and should therefore only be set for diagnostic purposes.

Output

Adapter for SAP logging output can be received by different log consumers. Currently Adapter for SAP 10.1 uses Integration Server logging mechanism to display the adapter log messages. You can configure and view the Integration Server logs to monitor and troubleshoot Adapter for SAP.

To receive and display the required logging information from the adapter, you need to set the log level of the SAP log facilities for Integration Server logging.

» To set the log level of the SAP log facilities for Integration Server logging

1. From Integration Server Administrator screen, select **Settings > Logging**. The **Logging Settings** screen appears.
2. In the **Logger List**, select **Server Logger**.
3. In the Server Logger Configuration section click on Adapter for SAP to display the current log level setting of the adapter's log facilities.
4. Click on **Edit Server Logger** to set the required log levels for Adapter for SAP facilities.
5. Click **Save Changes**.

For detailed information about logging in Integration Server, including instructions for configuring and viewing the different kinds of logs supported by the server, see *webMethods Integration Server Administrator's Guide* and *webMethods Audit Logging Guide* for your release.

Viewing and Deleting RFC Trace Files and SAP Log Files

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 Integration Server is running, you can view and delete RFC trace files and SAP log files from the IS Administrator UI.

All SAP log files, as well as the RFC trace files are stored in the packages/WmSAP/logs directory.

➤ To view and delete RFC trace files and SAP log files

1. In the IS Administrator UI, browse to **Adapters > Adapter for SAP > 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 **X** for the file to delete.
 - To quickly delete all trace files or all SAP log files, click **Delete All**.

Monitoring Statistics for Each Adapter for SAP Connection

The Monitoring screen allows the user to view the SAP repository connection pools, the SAP client connections, the SAP listener status, and the component response time monitor. From the Monitoring screen you may access a list of the size attributes of all currently active JCo Pools, as well as the attributes of all currently locked connections for a specific Adapter for SAP connection.

➤ To view Adapter for SAP connection statistics

1. In the **Adapters** menu in the Integration Server Administrator's navigation area, click **Adapter for SAP**.
2. In **Adapter for SAP** menu, click **Monitoring**. The Monitoring screen is displayed.
3. To view the size attributes of all currently active JCo Pools for a specific Adapter for SAP connection:

- a. In the SAP Client Connections section of the Monitoring screen, click the number in the **Pools** column that corresponds to the client connection for which you want to view the size attributes.
- b. The Connection Pools screen displays the following information:

Attribute	Description
Name	Name of the JCoPool.
Current size	Number of client connections that are currently open in the pool.
Peak size	Maximum number of client connections that were used simultaneously.
Max size	Maximum number of client connections that can be used simultaneously.
Current used	Number of client connections currently in use.

- c. To return to the Monitoring screen, click **Return to Monitoring**.
4. A locked connection is an RFC connection that is locked to your Integration Server session so that it will always be used for subsequent calls. An RFC connection is locked by calling the public Adapter for SAP service **pub.sap.client:lockSession**.

To view the attributes of all currently locked connections for a specific **Adapter for SAP** connection:

- a. In the SAP Client Connections section of the Monitoring screen, click the number in the **Locked Connections** column that corresponds to the client connection for which you want to view the attributes.
- b. The Locked Connections screen displays the following information:.

Attribute	Description
SAP user	SAP user name associated with this connection (empty if unavailable).
Client	Client number associated with this connection (empty if unavailable).
Language	Language associated with this connection (empty if unavailable).
Host	SAP host from where the connection is executed (empty if unavailable).

Attribute	Description
Conversation ID	Internal JCo conversation ID for the connection (empty if unavailable).
IS user	IS user name associated with the connection (empty if unavailable).

- c. To return to the Monitoring screen, click **Return to Monitoring**.

Monitoring Adapter for SAP Performance

Performance Output Information in the SAP Log File

From log level "Debug" or higher you can view performance output information for Adapter for SAP 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 Adapter for SAP service calls (ms)	Time needed for the invocation of a service to handle an inbound request in Adapter for SAP 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 Adapter for SAP 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 > Adapter for SAP > Monitoring**:

At the bottom of this screen, you will find the following 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.
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.

A Package Contents

■ Package Layout	248
------------------------	-----

Package Layout

Apart from the native libraries, all of the files relevant to Adapter for SAP and its persistent state information are in the `packages_directory\WmSAP` directory. The following table describes the directories and files that make up the SAP package directory.

Directory	Contents
code	Contains Adapter for SAP libraries, the JCo and SAP IDoc Class library jars, and sample source code.
config	Contains the <code>cbr.cnf</code> file, which is a configuration file that contains the configuration parameter for content-based routing.
logs	Contains Adapter for SAP log files, and RFC trace files.
ns	Contains namespace information for Adapter for SAP services.
pub, templates	Contains files and images comprising Adapter for SAP Administration user interface. Also contains all user and API documentation.
txStore	Contains the local transaction store with the transaction and transaction message body files.

B Adapter Configuration

■ General Adapter for SAP Settings	250
■ server.cnf	250

General Adapter for SAP Settings

For the general settings of Adapter for SAP, you can use Integration Server Administrator.

➤ To change the general settings

1. In the **Adapters** menu in Integration Server Administrator navigation area, click **Adapter for SAP**.
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.
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 Integration Server for the first time. Some of the following statements can be added manually after installing Adapter for SAP.

Use the **Edit Extended Settings** feature of Integration Server Administrator to view and edit the key statements. For more information about viewing and editing key statements, see *webMethods Integration Server Administrator's Guide* for your release.

Tip:

You can also add and edit key statements by shutting down Integration Server and manually editing the *Integration Server_directory\instances\instance_name\config\server.cnf* file.

The following statements are added automatically when Integration Server and Adapter for SAP 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.
watt.sap.listener. responseTime	Delay (seconds) until a listener responds at the latest to an incoming request.

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 white space 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

Setting	Description
	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.cacheFlush Period	Specifies how often, in seconds, the TransactionManager will flush unsaved changes in the transaction cache to the file system. Default: every 2 seconds. Maximum allowed value: 100 seconds.
watt.sap.xtn.cacheTimeTo Live	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 the number of seconds specified in 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. Default: 50 seconds. Maximum allowed value: 500 seconds.

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 debug log options:

Setting	Description
watt.sap.debug.CPICtrace	Sets the value(0.3) as JCo "cpi.trace" property. For more information, see SAP JCo documentation

Setting	Description
	It is not required to restart the adapter as the property value is updated each time when the WmSAP log level of the facility 102 JCo Messages is changed. Default: 0 = Off.
watt.sap.debug.sessionLog	Set to 'True' to enable detailed JCo session log messages. The adapter does not have to be restarted. Changes of this settings are detected automatically. Default: false.

Change/add the following statements to the server.cnf to specify transaction store options:

Setting	Description
watt.sap.xtn.cts.txstore	Sets the path to the external file directory for the Shared Transaction Store (STS). The directory must exist and it must have write access, otherwise the transaction store will not be enabled. Default: empty.
watt.sap.xtn.fileStore.threads	Sets the number of CPU threads to be used for listing the content of the transaction store and to execute the <code>pub.sap.transaction:list</code> service. Default: 3. Minimum: 1.
watt.sap.xtn.fallback	Set to 'yes' to let the adapter automatically use the local transaction store when the CTS setup is invalid or the CTS server is not available during adapter startup. Default: no

Change/add the following statements to the server.cnf to specify JCo options:

Setting	Description
watt.sap.jco.trace.dir	Sets the path to an existing and wireable external directory so that the JCo trace files will be stored in the external directory and not in the canonical subdirectory

Setting	Description
	IS/packages/WmSap/logs Default: empty.
watt.sap.jco.timeout	Sets the value during adapter startup as JCo 'jco.session_timeout' property to define the idle timeout for JCo session contexts in minutes. For more information, see SAP JCo documentation. Default: Content of "watt.server.clientTimeout", or 10 if this setting is not defined. Minimum: 1.
watt.sap.jco.clientConnectTimeout	Sets the value during adapter startup as JCo 'jrfc.client_connect_timeout' property to define the timeout for an RFC client logon or ping attempt in seconds. For more information, see SAP JCo documentation. Default: 60 Minimum: 1 Maximum: 3600

Change/add the following statements to the server.cnf to specify miscellaneous options:

Setting	Description
watt.sap.aclset	If this property is not available or not set to "true", then during the next startup of Integration Server, all Adapter for SAP-relevant ACL settings will be reset to their default values.
watt.sap.systat01.partnerNumber	To send SYSTAT01 IDocs from Adapter for SAP back to the SAP system, 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.
watt.sap.idoc.usePartnerProfileRelease	Set this property to true/yes for the SAP IDoc Class Library 1.0.8 or higher to remove unwanted fields. Setting the parameter to 'true' or 'yes' specifies that Adapter for SAP will not use IDoc release information while sending out IDocs. If you do not

Setting	Description
	<p>set the parameter (or set to any other value), then IDoc release information will be used for IDoc metadata lookup during transport.</p> <p>The parameter name is case-sensitive.</p>
watt.sap.systat01.idocLimit	<p>This property limits the maximum number of transactions related to IDoc processing that the pub.sap.monitor.systat01:report service will process and report on.</p> <p>Set the watt.sap.systat01.idocLimit parameter to specifying the maximum number of processed IDoc transactions that can be reported without slowing down Integration Server. A typical value could be 2000. The report service would then send SYSTAT01 IDocs for up to 2000 transactions only.</p>

Change/add the following statements to the server.cnf for backwards compatibility with previous Adapter for SAP version 4.6.

Setting	Description
watt.sap.compatibility	<p>This property enables compatibility with the WmPartners inboundProcess services. The WmPartners inboundProcess services automatically create a TID in the pipeline if there is none provided. If this property is set to "true" or "yes", the pub.sap.transport InboundProcess services will do the same to achieve compatibility with WmPartners inbound processing.</p>
watt.sap.ignoreCIMTYP	<p>If this property is set to "true", Adapter for SAP will not use the CIMTYP (extended IDoc type) for IDoc metadata lookup. This ensures backward compatibility for mapping of extended IDocs in applications developed before Adapter for SAP 4.6 SP5.</p>
watt.sap.RFC.use46WrapperTypes	<p>Adapter for SAP 4.6 encodes numeric and binary RFC data fields as String objects in the pipeline, while Adapter for SAP 6.5 (or higher) uses Java objects to encode numeric and binary RFC data. If this property is set to "true" or "yes", Adapter for SAP 10.1 will use the same RFC data encoding as Adapter for SAP 4.6 to achieve compatibility with mappings in existing applications for Adapter for SAP 4.6.</p>

C ABAP Types in Adapter for SAP

■ ABAP Types	258
--------------------	-----

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	FLTP	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, LRAW	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[]

D Built-in Services

■ SAP Client Services	260
■ XRFC Services	272
■ IDoc Services	274
■ IDoc-XML Services	280
■ Monitoring Services	283
■ bXML Services	284
■ BAPI Services	286
■ Transaction Administration Services	287
■ Transport Services	292
■ Specifications	302
■ Sample Services	306
■ SAP Listener Services	309
■ SAP Utility Services	311
■ webMethods Adapter for SAP IDoc Java API	311

SAP Client Services

pub.sap.client:connect

Establishes a connection to an SAP system.

Input Parameter

serverName	Alias of the SAP system to which the connection is established. The name must match a configured RFC connection alias at Adapter for SAP.
\$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 the password is not specified, then the user and password will be looked up in the SAP User store.
\$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.

x

Input Parameter

serverName	Alias of the SAP system to which the connection is established. The name must match a configured RFC connection alias at Adapter for SAP.
\$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 the password is not specified, then the user and password will be looked up in the SAP User store.
\$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

None

Example

Note:

This service locks a session to trigger a `commit work` command inside the SAP system. This causes a database commit. 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 and the start of the processing of posted data.

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 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 Integration Server.

pub.sap.client:releaseSession

Releases a locked session on an SAP system.

Input Parameter

serverName	SAP system alias on which the locked session is located. The name must match a configured RFC connection alias at Adapter for SAP. .
\$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 the password is not specified, then the user and password will be looked up in the SAP User store.

`$language` Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

None

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 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 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 Parameter

<code>serverName</code>	SAP system alias on which the function module will be invoked. The name must match a configured RFC connection alias at Adapter for SAP.
<code>\$rfcnamep</code>	Name of the function module to be invoked.
<code>\$client</code>	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
<code>\$user</code>	Optional. User name for the session. If no user is specified, the default user is used.
<code>\$pass</code>	Optional. Password for the session. If the password is not specified, then the user and password will be looked up in the SAP User store.
<code>\$language</code>	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.

<code>\$rfctime</code>	Time (ms) spent within the RFC library to complete the invocation.
<code>\$runtime</code>	Total invocation time (ms), including processing time within Adapter for SAP.
<code>\$encoding</code>	MIME-compliant character set corresponding to the session's SAP code page.
<code>\$call</code>	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 Parameter

<code>serverName</code>	SAP system alias on which the function module will be invoked. The name must match a configured RFC connection alias at Adapter for SAP.
<code>\$rfcnamep</code>	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 the password is not specified, then the user and password will be looked up in the SAP User store.
\$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 Adapter for SAP.
\$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 or response of a function module. Possible values are: <ul style="list-style-type: none">■ true for request.■ false for response.

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`.

For more information about using the fast, offline method of creating Transaction ID, see [“watt.sap.fastTIDCreation” on page 314](#).

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 Parameter

<code>serverName</code>	Alias of the SAP system that sends the TID. This name must match a configured RFC connection alias at Adapter for SAP.
<code>\$client</code>	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
<code>\$user</code>	Optional. User name for the session. If no user is specified, the default user is used.
<code>\$pass</code>	Optional. Password for the session. If the password is not specified, then the user and password will be looked up in the SAP User store.
<code>\$language</code>	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 Parameter

<code>serverName</code>	Alias of the SAP system that sends the TID. This name must match a configured RFC connection alias at Adapter for SAP.
<code>\$tid</code>	Transaction ID to be confirmed.
<code>\$client</code>	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.

<code>\$user</code>	Optional. User name for the session. If no user is specified, the default user is used.
<code>\$pass</code>	Optional. Password for the session. If the password is not specified, then the user and password will be looked up in the SAP User store.
<code>\$language</code>	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 is executed completely.

1. Invoke `clientCreateTID`. This service creates an SAP-conformant TID. It is necessary to have one, otherwise you cannot invoke the function module with `clientInvokeTransaction`. If you have already obtained a TID, you must use that one in order to guarantee that the transaction is executed only once.
2. Invoke `clientInvokeTransaction`. 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 `clientConfirmTID`. Confirms that the transaction is 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 Parameter

<code>serverName</code>	Alias of the SAP system that sends the TID. This name must match a configured RFC connection alias at Adapter for SAP.
<code>\$client</code>	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
<code>\$user</code>	Optional. User name for the session. If no user is specified, the default user is used.
<code>\$pass</code>	Optional. Password for the session. If the password is not specified, then the user and password will be looked up in the SAP User store.
<code>\$language</code>	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

<code>destination</code>	String. Destination of the connection.
<code>ownHost</code>	Hostname (or IP) of the machine Integration Server is running on.
<code>partnerHost</code>	Hostname (or IP) of the machine the SAP system is running on.
<code>systemNumber</code>	System number of the SAP system.
<code>systemID</code>	Unique three-letter-ID of SAP system (in local network).
<code>client</code>	Client the session is connected to.
<code>user</code>	SAP user that has connected with this session.
<code>language</code>	Logon language.
<code>SOLanguage</code>	ISO name for logon language.
<code>ownCodepage</code>	SAP codepage this connection is using.
<code>partnerCodepage</code>	SAP codepage the SAP system is running on.
<code>ownRelease</code>	Version of the loaded RFC library used by Adapter for SAP.
<code>partnerRelease</code>	Release of the SAP system.
<code>kernelRelease</code>	Kernel release of the SAP system.
<code>partnerType</code>	RFC type of the partner; SAP ECC(3), R/2(2) or external RFC server(E).
<code>trace</code>	Flag indicating whether trace is turned on (true) or off (false) for this connection.
<code>ownType</code>	RFC type of Adapter for SAP; is always be E.
<code>rfcRole</code>	Role of Adapter for SAP in this call; is always be: C (for client).
<code>CPICConversationID</code>	CPIC ID of the connection (low level protocol information).
<code>encoding</code>	IANA-encoding that is equivalent to the SAP codepage used, for example: ISO-8859-1.
<code>charset</code>	Charset that is equivalent to the SAP codepage used, for example: ISO8859_1 (used in Java constructors).
<code>bytesPerChar</code>	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 Parameter

<code>\$rfcname</code>	Function module for which the interface definition is looked up.
<code>serverName</code>	Alias of the SAP system on which the RFC function module is defined. The name must match a configured RFC connection alias at Adapter for SAP.

Return Values

Document list named *params* that contains all parameter-related information:

<code>paramClass</code>	Class of parameter; I (importing), E (exporting), T (table), X (exception).
<code>parameter</code>	Name of the parameter in the function interface.
<code>tabName</code>	Name of the table in the SAP system the parameter is referring to.
<code>type</code>	Type of the parameter (for example: CHAR, STRUCTURE, TABLE, ...)
<code>length</code>	Internal length in bytes of the parameter.
<code>decimals</code>	Only relevant for decimal types; number of decimals used.
<code>optional</code>	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 Parameter

<code>structName</code>	Structure name for which the definition is looked up.
<code>serverName</code>	Alias of the SAP system on which the RFC function module is defined. The name must match a configured RFC connection alias at Adapter for SAP.

Return Values

Document list named *params* that contains all parameter-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:getThroughput

Returns the performance throughput value of the connection since the adapter startup or the last reset to zero

Input Parameter

serverName	Alias for the SAP system from which you want to retrieve the RFC connection specific attributes. The name must match a configured RFC connection alias at Adapter for SAP. Optional. Set to true to reset the performance throughput values to zero
\$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 the password is not specified, then the user and password will be looked up in the SAP User store.
\$language	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

numberOfCalls	Number of calls executed by the connection.
numberOfBytesSent	Number of bytes sent by the connection.
numberOfBytesReceived	Number of bytes received by the connection.
marshallingTime	Time spent by the connection to convert pipeline data into RFC data (in ms).

<code>unmarshallingTime</code>	Time spent by the connection to convert RFC data into pipeline data (in ms).
<code>timeForPreparation</code>	Time spent by the connection for preparation and repository lookups (in ms).
<code>timeForMiddlewareCalls</code>	Time spent by the connection during RFC network calls(in ms).
<code>timeForAdapterServiceCalls</code>	Time spent by the connection for processing inbound or outbound requests in Integration Server services.
<code>totalTime</code>	Total time spent by the connection for processing inbound and outbound services.

pub.sap.client:sendIDoc

Sends an IDoc list to a given SAP system. The IDocs in the list must be of the same IDoc type.

Input Parameter

<code>serverName</code>	Alias for the SAP system from which you want to retrieve the RFC connection specific attributes. The name must match a configured RFC connection alias at Adapter for SAP. Optional. Set to true to reset the performance throughput values to zero
<code>iDocList</code>	Contains the IDoc(s) as object of the <code>com.sap.conn.idoc.IDocDocumentList</code> type.
<code>\$tid</code>	The ID of the transaction under which the service is executed
<code>\$queueName</code>	Optional. Name of the SAP system IDoc inbound queue. Specify a value in case of an ALE queuing scenario.
<code>\$client</code>	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
<code>\$user</code>	Optional. User name for the session. If no user is specified, the default user is used.
<code>\$pass</code>	Optional. Password for the session. If the password is not specified, then the user and password will be looked up in the SAP User store.
<code>\$language</code>	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

<code>\$runtime</code>	Total invocation time (ms), including processing time within Adapter for SAP.
<code>\$rfcTime</code>	Time (ms) spent within the RFC library to complete the invocation.
<code>\$encoding</code>	MIME-compliant character set corresponding to the session's SAP code page.

<code>\$call</code>	Flag indicating whether pipeline represents a request or response of a function module. Possible values are:
■	true for request.
■	false for response.

pub.sap.client:sendIDocLists

Sends a heterogeneous list of IDoc lists to a given SAP system. All the IDoc elements in an IDoc list must be of the same IDoc type, but the IDoc lists can contain different types. All IDocs can be processed in a single transaction.

Note:

Install webMethods Adapter for SAP Fix 1 and later to use this service.

Input Parameter

<code>serverName</code>	Alias for the SAP system from which you want to receive the RFC connection specific attributes. The name must match a configured RFC connection alias at Adapter for SAP. Optional. Set to true to reset the performance throughput values to zero
<code>iDocList</code>	Contains the IDoc(s) as object of the <code>com.sap.conn.idoc.IDocDocumentList</code> type.
<code>\$tid</code>	ID of the transaction under which the service is executed.
<code>\$queueName</code>	Optional. Name of the SAP system IDoc inbound queue. Specify a value in case of an ALE queuing scenario.
<code>\$client</code>	Optional. Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
<code>\$user</code>	Optional. User name for the session. If no user is specified, the default user is used.
<code>\$pass</code>	Optional. Password for the session. If the password is not specified, then the user and password will be looked up in the SAP User store.
<code>\$language</code>	Optional. Language used during the session. If no language is specified, the default language is used.

Return Values

<code>\$runtime</code>	Total invocation time (ms), including processing time within Adapter for SAP.
<code>\$rfctime</code>	Time (ms) spent within the RFC library to complete the invocation.
<code>\$encoding</code>	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 Parameter

`serverName` Alias for the SAP system from which you want to receive the RFC connection specific attributes. The name must match a configured RFC connection alias at Adapter for SAP. Optional. Set to true to reset the performance throughput values to zero

`$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 the password is not specified, then the user and password will be looked up in the SAP User store.

`$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 Parameter

`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 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 Parameter

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 Adapter for SAP.

`$encoding` Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1

`$call` Flag indicating whether pipeline represents a request or response of a function module. Possible values are:

- true for request.
- false for response.

`$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 Parameter

serverName	SAP system used as a repository. This name must match a configured RFC connection alias at Adapter for SAP.
\$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	Flag indicating whether pipeline represents a request or response of a function module. Possible values are: <ul style="list-style-type: none">■ true for request.■ false for response.
\$envelope	Optional. XRFC, bXML, or SOAP. Specifies the type of document 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

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](#)" on [page 270](#) that allows you to send the iDocList directly.

Input Parameter

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 Adapter for SAP.
IDOC_CONTROL	Both keys are Document lists (Tables) containing the control and data tables for the IDoc. 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).
IDOC_DATA	
- OR -	
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.	
iDocList	Contains the IDoc(s) as object of com.sap.conn.idoc.IDocDocumentList.
- OR -	
\$encoding	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.

Return Values

IDOC_CONTROL	Both keys are document lists (Tables) containing the control and data tables for the IDoc.
IDOC_DATA	
- OR -	The SDATA field is a 1000-byte field.
IDOC_CONTROL_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").
IDOC_DATA_REC40	

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.conn.idoc.IDocDocumentList` object for further processing.

With Adapter for SAP 6.5 or higher, 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 Parameter

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 Adapter for SAP.
IDOC_CONTROL	Both keys are Document lists (Tables) containing the control and data tables for the IDoc.
IDOC_DATA	
- OR -	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).
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.conn.idoc.IDocDocumentList`.

pub.sap.idoc:encodeString

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 `clientInvokeTransaction` or by an RFC adapter service for function module `IDOC_INBOUND_ASYNCHRONOUS` or `INBOUND_IDOC_PROCESS`.

Note:

A client service “[pub.sap.client:sendIDoc](#)” on [page 270](#) allows you to send the `iDocList` directly.

Input Parameter

<code>serverName</code>	SAP system alias which is used as repository for structure information about the IDoc. The name must match a configured RFC connection alias at Adapter for SAP.
<code>iDocRelease</code>	Optional. The SAP system release that the IDoc will be sent to.
<code>\$encoding</code>	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.
<code>iDocList</code> - OR -	Contains the IDoc(s) as object of <code>com.sap.conn.idoc.IDocDocumentList</code> .
<code>IDOC_CONTROL</code> <code>IDOC_DATA</code> - OR - <code>IDOC_CONTROL_REC40</code> <code>IDOC_DATA_REC40</code>	Both keys are Document lists (Tables) containing the control and data tables for the IDoc. 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 <code>SEGNAM</code> 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 <code>IDOC_CONTROL</code> and <code>IDOC_DATA</code> in the pipeline. For IDocs version 3, it looks for <code>IDOC_CONTROL_REC_40</code> and <code>IDOC_DATA_REC_40</code> .	

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 Parameter

<code>serverName</code>	SAP system alias which is used as repository for structure information about the IDoc. The name must match a configured RFC connection alias at Adapter for SAP.
<code>iDocString</code>	String. The iDoc in the file port.
<code>\$encoding</code>	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.conn.idoc.IDocDocumentList`.

pub.sap.idoc:iDocToDocument

Generates a Document representation of an IDoc list to perform mappings in Flow language.

Input Parameter

<code>iDocList</code>	Contains the IDoc(s) as an object of the type <code>com.sap.conn.idoc.IDocDocumentList</code> .
<code>conformsTo</code>	Name of the document structure of the IDoc. The document structure is used as a template to discriminate between documents and document lists. Without <code>conformsTo</code> , 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 Parameter

`document` IDoc in hierarchical structure.

Return Values

`iDocList` Contains the IDoc(s) as object of `com.sap.conn.idoc.IDocDocumentList`.

pub.sap.idoc:iDocToTables

Generates the tables compatible with Adapter for SAP version 4.6 from an object compatible with Adapter for SAP version 10.1 or higher.

Do this before sending the IDoc to Adapter for SAP version 4.6 inbound transport.

Input Parameters

`iDocList` Contains the IDoc(s) as object of `com.sap.conn.idoc.IDocDocumentList`.

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 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.

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 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.

Return Values

`iDocList` Contains the IDoc(s) as object of `com.sap.conn.idoc.IDocDocumentList`.

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	Contains the IDoc(s) as object of the type com.sap.conn.idoc.IDocDocumentList.
- OR -	
IDOC_CONTROL	Both keys are Document lists (Tables) containing the control and data tables for the IDoc.
IDOC_DATA	
- OR -	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).
IDOC_CONTROL_REC40	
IDOC_DATA_REC40	
<p>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.</p>	
\$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.conn.idoc.IDocDocumentList` 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 <code>com.sap.conn.idoc.IDocDocumentList</code> .

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.conn.idoc.IDocDocumentList` 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.conn.idoc.IDocDocumentList` 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 Adapter for SAP.

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.

<code>\$encoding</code>	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.
<code>\$call</code>	Optional. Flag indicating whether pipeline represents a request or response of a function module. Possible values are: <ul style="list-style-type: none"> ■ <code>true</code> for request. ■ <code>false</code> for response.
<code>\$abapexception</code>	Optional. If this object is in the pipeline, an Exception document will be created.
<code>\$metabapi</code>	Optional. Contains metadata information about the BAPI, that is 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

<code>serverName</code>	SAP system alias for the SAP system that is used as a repository. The alias must match a configured RFC connection alias at Adapter for SAP.
<code>objectName</code>	Name of the business object to encode.
<code>bapiName</code>	Name of the BAPI.
<code>\$encoding</code>	Optional. Specifies the encoding used in the output document's XML header, e.g. iso-8859-1.
<code>\$call</code>	Optional. Flag indicating whether the pipeline is interpreted as a call or as a response. Possible Values are: <ul style="list-style-type: none"> ■ <code>true</code> for call. ■ <code>false</code> for response. Default.

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 is committed. This name must match an RFC connection alias at Adapter for SAP.
\$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 the password is not specified, then the user and password will be looked up in the SAP User store.
\$language	Optional. Language used during the session. If no language is specified, the default language is used.
wait	Optional. Flag indicating if call will 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 keep your connection in transaction scenarios because the transaction context is stored with the session in the SAP system.

Input Parameters

serverName	Alias of the SAP system on which the transaction is committed. This name must match an RFC connection alias at Adapter for SAP.
\$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 the password is not specified, then the user and password will be looked up in the SAP User store.
\$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 that match the filter criteria. Regular expressions are supported to allow flexible filter criteria matching. For more information of regular expressions, see *webMethods Service Development Help* for your release.

Input Parameters

endDate	Optional. Filter criteria: Last creation date of the transaction. The format of this field is 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 is 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.
\$tid	Optional. Filter criteria: TID of the transaction.
\$sortKey	Optional. Specifies the sort key - valid: <ul style="list-style-type: none"> ■ noSort. Default. ■ \$tid ■ sender ■ receiver

- msgType
- date
- state

The resulting transaction list is sorted by the given attribute.

\$dir	Optional. Specifies the sort direction. Possible values: <ul style="list-style-type: none">■ descending. Triggers a descending sort order.■ Any value other than descending. 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: <ul style="list-style-type: none">■ \$tid■ sender■ receiver■ msgType■ date. Date/time, when the state of this transactions was last changed, formatted in the usual "Log Timestamp Format"■ state
--------------	--

\$sortKey	Optional. Specifies the sort key - valid: <ul style="list-style-type: none">■ noSort. Default.■ \$tid■ sender■ receiver■ msgType■ date■ state <p>The resulting transaction list is sorted by the given attribute.</p>
-----------	---

\$dir	Optional. Specifies the sort direction. Possible values: <ul style="list-style-type: none">■ descending. Triggers a descending sort order.
-------	--

-
- Any value other than descending. 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

<code>\$tid</code>	The TID of the transaction.
<code>msg</code>	Optional. Message string to be logged.
<code>state</code>	Optional. String . Valid values are: <ul style="list-style-type: none">■ Created■ Executed■ Committed■ Rolledback■ Confirmed

Return Values

None.

pub.sap.transaction:setCacheParameter

This service sets new runtime parameters for the cache of the transaction store.

Input Parameters

<code>writeCache</code>	Optional. State of the write cache. Possible values are: <ul style="list-style-type: none">■ on. Write cache of the transaction is switched on.■ Any other value. Write cache of the transaction is switched off.
<code>cacheTimeToLive</code>	Optional. This value sets the maximum time-to-live for a cache entry (value in seconds). Valid range: 0-500.
<code>cachFlushPeriod</code>	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	"FileTransactionStore at <directory>" for Local or Shared Transaction Store or "Remote Tx Store" for CTS.
writeCache	State of the cache. Possible values are: <ul style="list-style-type: none"> ■ On ■ Off
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

<code>elapsedTime</code>	Filter criteria: time in minutes since the transaction has been modified the last time.
<code>maxTrxCount</code>	Optional. Maximum number of transactions to be deleted in one execution of the Service.
<code>sender</code>	Optional. Filter criteria: name of the sender of the transaction.
<code>receiver</code>	Optional. Filter criteria: name of the receiver of the transaction.
<code>msgType</code>	Optional. Filter criteria: name of the message type of the transaction.
<code>state</code>	Optional. Filter criteria: state of the transaction. Valid values are: <ul style="list-style-type: none">■ Created■ Executed■ Committed■ Rolledback■ Confirmed

For more information about cleaning the transaction store, see [“Automatic Cleanup of the Transaction Stores” on page 181](#).

Return Values

<code>deletedEntries</code>	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 <code>com.sap.conn.idoc.IDocDocumentList</code>
\$tid	Optional. The TID of the transaction.
\$action	Optional. One of the following transaction codes: <ul style="list-style-type: none"> ■ 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

serverName	Optional. Alias of the SAP system to which the connection is established. The name must match a configured RFC connection alias at Adapter for SAP. If serverName is set, it will override the value of <code>transportParams/serverName</code> .		
transportParams	A document with the following key/value pair: <table border="1"> <tr> <td>serverName</td><td>SAP system alias for the SAP system on which invoke the RFC. The alias must match a configured RFC connection alias at Adapter for SAP.</td></tr> </table>	serverName	SAP system alias for the SAP system on which invoke the RFC. The alias must match a configured RFC connection alias at Adapter for SAP.
serverName	SAP system alias for the SAP system on which invoke the RFC. The alias must match a configured RFC connection alias at Adapter for SAP.		
iDocList	Optional. Contains the IDoc(s) as an object of the type <code>com.sap.conn.idoc.IDocDocumentList</code> .		
\$tid	Optional. The TID of the transaction.		
\$action	One of the following transaction codes: <ul style="list-style-type: none"> ■ 1 - Execute (default) ■ 4 - Confirm <p>Run the service in the following order:</p>		

1. With `$action` set to 1 or empty to send the IDoc to the SAP system.
2. For the same value of `$tid` and `$action` set to 4 to inform the SAP system that the transaction is complete.

<code>\$queueName</code>	Optional. Name of the SAP system IDoc inbound queue. Specify a value in case of an ALE queuing scenario.
--------------------------	--

Return Values

<code>\$runtime</code>	Total invocation time (ms), including processing time within Adapter for SAP.
------------------------	---

<code>\$rfctime</code>	Time (ms) spent within the RFC library to complete the invocation.
------------------------	--

<code>\$encoding</code>	MIME-compliant character set corresponding to the session's SAP code page.
-------------------------	--

<code>\$call</code>	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

<code>iDocList</code>	Contains the IDoc(s) as an object of the type <code>com.sap.conn.idoc.IDocDocumentList</code> .
-----------------------	---

Return Values

<code>iDocList</code>	Contains the IDoc(s) as an object of the type <code>com.sap.conn.idoc.IDocDocumentList</code> .
-----------------------	---

<code>sender</code>	Retrieved from sender partner number from the header of the first IDoc.
---------------------	---

<code>receiver</code>	Retrieved from receiver partner number from the header of the first IDoc.
-----------------------	---

<code>msgType</code>	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 Adapter for SAP. Use a key for the message that the outbound transport expects.

sender	Sender of a message.
receiver	Receiver of the message.
msgType	Message type of the message.
\$tid	Optional. TID of the transaction.
\$action	Optional. One of the following transaction codes: <ul style="list-style-type: none"> ■ 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	<p>Optional. Alias of webMethods Integration Server on which the service is to be invoked.</p> <p>To invoke a service on a remote server:</p> <ul style="list-style-type: none">■ 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. Transaction ID. Specify a transaction ID if the document should be sent as a transaction.
\$action		<p>Optional. One of the following transaction codes:</p> <ul style="list-style-type: none">■ 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 Adapter for SAP.

Input Parameters

This service requires the following input parameters:

sbchHeader 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 151](#).

\$rfcname Optional. The name of the function module.

\$tid Optional. 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. Transaction state. Specify one of the following codes:

- 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:

serverName Optional. Alias of the SAP system to which the connection is established. The name must match a configured RFC connection alias at Adapter for SAP.

If **serverName** is set, it will override the value of **transportParams/serverName**.

transportParams A document with the following key/value pair.

Key	Description
serverName	SAP system alias for the SAP system on which invoke the RFC. The alias must match a configured RFC connection alias at Adapter for SAP.
\$tid	Optional. 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. Transaction state. Specify one of the following codes: <ul style="list-style-type: none">■ 1 - Execute (default)■ 4 - Confirm
\$rfcname	Optional. RFC that is called.
\$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 Adapter for SAP.
\$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

This service requires the following input parameters:

node	XML Node object that can be generated by calling <code>pub.xml:xmlStringToXMLNode</code> and is automatically generated when posting an XML document via HTTP (using text/xml).
\$tid	Optional. TID of the transaction.
\$action	Optional. Transaction state. Specify one of the following codes: <ul style="list-style-type: none">■ 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:

`transportParams` A document with the following key/value pair.

Key	Description
<code>url</code>	The URL to which to post the document.
<code>xmlType</code>	<p>The XML dialect to use. Specify <code>values-XML</code> if you want the XML in webMethods native XML format.</p> <ul style="list-style-type: none"> Specify <code>SAP-XML</code> 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).
<code>useTextXml</code>	Optional. Flag that overwrites the content-type of bXML, XRFC and IDoc-XML to text/xml, if set to true.
<code>useUTF8</code>	Optional. Flag that allows to force the XML rendering engines to use UTF-8 as encoding for the resulting documents.
<code>httpUser</code>	Optional. User name to supply for a user name/password authentication challenge.
<code>httpPassword</code>	Optional. Password to supply along with <code>httpUser</code> for a user name/password authentication challenge.
<code>useBAPI</code>	ON or OFF. If set to ON, the pipeline is encoded using bXML for Business Objects.
<code>objectName</code>	Name of the business object, to which the call is mapped.

	<code>bapiName</code>	Name of the BAPI method, to which the call is mapped.
<code>xmlData</code>	Optional. An XML document as string, only used if <code>transportParams/xmlType</code> is <code>Arbitrary XML</code> .	
<code>\$encoding</code>	Optional. Encoding used by the XML request document.	
<code>\$tid</code>	Optional. Transaction ID. Specify a transaction ID if the document should be sent as a transaction.	
<code>\$action</code>	Optional. Transaction state. Specify one of the following codes: <ul style="list-style-type: none">■ 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

<code>sbcHeader</code>	Document that contains the key/value pairs with routing information. Will be generated by the bXML parser.
<code>objectName</code>	Optional. Name of the business object that is represented in the pipeline.
<code>bapiName</code>	Optional. Name of the BAPI that is represented in the pipeline.
<code>\$action</code>	Optional. Transaction state. Specify one of the following codes: <ul style="list-style-type: none">■ 1 - Execute (default)■ 4 - Confirm
<code>\$tid</code>	Optional. 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

serverName	Optional. Alias of the SAP system to which the connection is established. The name must match a configured RFC connection alias at Adapter for SAP. If serverName is set, it will override the value of transportParams/serverName.		
transportParams	A document with the following key/value pair.		
	Key	Description	
	serverName	SAP system alias on which to execute the BAPI. The alias must match a configured RFC connection alias at Adapter for SAP.	
	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. TID of the transaction.		
\$action	Optional. Transaction state. Specify one of the following codes:		
	<ul style="list-style-type: none"> ■ 1 - Execute (default) ■ 4 - Confirm 		

<code>\$queueName</code>	Optional. Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario.
<code>objectName</code>	Optional. Name of the business object.
<code>bapiName</code>	Optional. Name of the BAPI method.

Return Values

A pipeline that represents the response or an exception of the executed BAPI.

<code>\$metabapi</code>	Optional. Document created if the BAPI was executed. Contains the BAPI metadata (for example, the internal name, the external name, and the RFC name). It also contains information about whether it is static (yes/no), dialog (yes/no), or factory BAPI (yes/no).
<code>\$abapexception</code>	Optional. Document created if the execution of the BAPI caused one or more ABAP exceptions with severity of 3 or higher. The document contains the name, the severity, and the message attributes of the general BAPI failure. It also contains the document attributes that contain the complete collection of all exceptions that have occurred during execution of the BAPI, with a detailed description and the message attributes for each exception.
<code>\$runtime</code>	Total invocation time (ms), including processing time within Adapter for SAP.
<code>\$rfctime</code>	Time (ms) spent within the RFC library to complete the invocation.
<code>\$encoding</code>	MIME-compliant character set corresponding to the session's SAP code page.
<code>\$call</code>	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 is 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	Function that has been called by a remote client.
partnerName	Partner (system) name where the request comes from.
key	Authorization key as binary data depending on the mode.
mode	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	Returns "granted" if the request is accepted.
--------	---

pub.sap.transport.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.conn.idoc.IDocDocumentList.
----------	---

Return Values

iDocList	Contains the IDoc(s) as an object of the type com.sap.conn.idoc.IDocDocumentList.
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.transport.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 <code>com.sap.conn.idoc.IDocDocumentList</code> .
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 <code>com.sap.conn.idoc.IDocDocumentList</code> .
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.transport.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 <code>com.sap.conn.idoc.IDocDocumentList</code> .
----------	---

Return Values

iDocList	Contains the IDoc(s) as an object of the type <code>com.sap.conn.idoc.IDocDocumentList</code> .
msgType	Contains the message type value as determined by this ALE default routing info service.

pub.sap.transport.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.conn.idoc.IDocDocumentList`.

`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.conn.idoc.IDocDocumentList`.

pub.sap.transport.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` 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:

- 1 - Execute (default)
- 4 - Confirm

Sample Services

sample.sap:handleIDocXMLPost

Submit an IDoc-XML message to ALE InboundProcess on Adapter for SAP.

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:

- 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 Adapter for SAP.

`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	The alias for the SAP system on which to execute the RFC. The alias must match a configured RFC connection alias at Adapter for SAP.	
xmlData	String. The XML message of the RFC-XML document.	
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:returnABAPException

This service returns a JCO AbapException exception that is used by the service implemented at Integration Server. By adding an invoke statement to the sample service you can easily return an ABAP exception from Flow. The sample also demonstrates how to return an ABAP exception from within a Java service.

Input Parameters

key The key for the exception that is specified in the function interface.

message Optional. Additional descriptive error message.

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 Adapter for SAP.

mode Allows you to choose the way how the BAPI will be invoked. Possible values are:

- **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 Adapter for SAP.

\$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.transport.ALE:aleRoutingInfo” on page 303](#).

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.transport.ALE:aleMappingInfo” on page 304](#).

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 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.
SOLanguage	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 Adapter for SAP.
partnerRelease	Release of the SAP system.
kernelRelease	Kernel release of the SAP system.
partnerType	RFC type of the partner; SAP ECC(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 Adapter for SAP; is always be E.
rfcRole	Role of Adapter for SAP in this call; is 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.

SAP Utility Services

pub.sap.util: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 even when not connected to a SAP system. The Transaction ID is based on a randomly created UUID Version 4.

The obtained TID can be used as an input value for `pub.sap.client:invokeTransaction`, `pub.sap.client:sendIDoc`, and `pub.sap.client:confirmTID`.

Input Parameters

None.

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`.

webMethods Adapter for SAP IDoc Java API

See `packages_directory\WmSAP\pub\doc\api\index.html`.

E Configuration Parameters

■ watt.sap.fastTIDCreation	314
----------------------------------	-----

watt.sap.fastTIDCreation

Configures Adapter for SAP to use the fast, offline method of creating Transaction ID in the existing public service `pub.sap.client:createTID`. Default value is `false`.

You can configure the following property on the **Extended Settings** screen (**Settings > Extended**) in Integration Server Administrator:

```
watt.sap.fastTIDCreation=true
```

F Deprecated Services

■ List of Deprecated Services	316
-------------------------------------	-----

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 Adapter for SAP Administration UI instead.
pub.sap.idoc.routing:unregisterService	Use Adapter for SAP Administration 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.

G Working with Code Pages

■ Using Different Code Pages	318
------------------------------------	-----

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, or 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 Adapter for SAP with proper encoding.

To Encode Data from SAP Systems

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".

H Using BizTalk Envelopes with Adapter for SAP

■ Overview	322
■ Use of the BizTalk Header	322
■ Error Handling	325

Overview

Adapter for SAP uses the BizTalk XML envelope as the transport envelope for the transmission of XML business documents. The BizTalk envelope is implemented on Adapter for SAP 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 Adapter for SAP 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>xyzy: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*.

Adapter for SAP only uses a subset of these available header elements, which is described in the following chapters.

Representation of Routing and Address Information

Adapter for SAP 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 Adapter for SAP. 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>0A125F1315B3A11B000000035</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), Adapter for SAP 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>0A125F1315B3A11B000000035</messageID>
        <sent>2000-06-07T09:22:40</sent>
      </message>
      <to>
        <address>urn:sap-com:logical-system:SAPCLNT001</address>
        <state>
          <referenceID>120A135FB315A11B000003500</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. Adapter for SAP will always generate a new message ID for both request and response messages.
<sent>	<p>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.</p> <p>For example:</p> <p>2000-06-19T18:59:02describes June 19, 2000 18 hours 59 minutes 2 seconds.</p>
<subject>	<p>May be used to specify an additional description for the message.</p> <p>Important: This element is not used by Adapter for SAP.</p>

Important:

BizTalk provides the possibility to exchange manifest information in the header. This feature is not used by Adapter for SAP.

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.

Adapter for SAP distinguishes between two major groups of errors:

- Errors in the XML processing and conversion layers of Adapter for SAP, 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” on page 326](#)

■ [“Representation of Application Errors” on page 327](#)

Representation of Communication and Processing Errors

Errors which are caused inside Adapter for SAP 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
<code><faultcode></code>	A number specifying the class of error. Compatible with the Microsoft SOAP XML framework specification. Adapter for SAP always sets this value to 401 to indicate an application-specific error.
<code><faultstring></code>	Internal Adapter for SAP code for this exception.
<code><detail></code>	Details of the specified sub-elements <code><name></code> and <code><message></code>
<code><name></code>	Name of an exception to use with the SAP system as the ABAP exception identifier.
<code><message></code>	Error message with the specified sub-element <code><text></code> . (Optional; this is not always fully specified).
<code><text></code>	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>0A125F1315B39EDB000000013</messageID>
        <sent>2000-06-19T18:59:02</sent>
      </message>
    </delivery>
    <to>
      <address>urn:sap-com:logical-system:SAPCLNT001</address>
    </to>
    <from>
      <address>urn:sap-com:logical-system:SAPADA0001</address>
    </from>
  </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 [“Using IFR-XML Format with Adapter for SAP” on page 329](#). 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.

I Using IFR-XML Format with Adapter for SAP

■ Overview	330
■ XML Format for BAPIs	332
■ XML Format for RFCs	337
■ XML Format for IDocs	339

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, ...).

Adapter for SAP 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 [“Using BizTalk Envelopes with Adapter for SAP” on page 321](#).

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*.

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.
<AttributenameX> (multiplicity: 0..*)	Each serialized exception within the <Collection> element has the same structure as the wrapper exception.
	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*.

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.MethodeName.Response
  xmlns:doc="urn:sap-com:document:sap:business"
  [Key1="...", Key2="...", ...]>
  <BOR_Parameter1>...</BOR_Parameter1>
  ...
</doc:BusinessObjektName.MethodeName.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*.
- 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“.

XML Element	Corresponding Information from the Return Parameter
	„BapiAbort“, if value of field TYPE is „A“.
<Message> (0..1)	Value of the ID field.
<ID>	Value of the NUMBER field.
<Number>	Value of the MESSAGE field.
<Text>	
<Attributes> (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.
<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)	

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*. 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.
<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) ...	The <Status> element contains a serialized table with row type BAPIRET2. Within the <Status>

XML Element	Corresponding Information from the Return Parameter
<code></item></code> <code>...</code> <code></Status></code>	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*.
- 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*.

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> <Number> <Text>	SY-MSGID SY-MSGNO 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.

XML Element	Description
<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.