webMethods SAP Adapter

User's Guide

VERSION 4.6

webMethods, Inc.
3930 Pender Drive
Fairfax, VA 22030
USA
703.460.2500
http://www.webmethods.com

webM-SAP-UG-20020430

Typographical Conventions

Type Style	Description
Interface Text	Words or characters that appear on the screen. This includes field names, screen titles, pushbuttons, menu names, and menu options.
Document Title	Cross-references to other documentation.
`User Entry`	Words and characters that you enter exactly as they appear in the documentation.
<Variable User Entry> or *Variable User Entry*	Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries.
KEY	Keys on the keyboard. These include the function keys, e.g. **F2,** or **ENTER**.

Icon	Description
➡	Note containing important information (for example, exceptions or special cases).
⚠	Caution that helps users avoid errors which can lead to problems such as loss of data.
💡	Hints and tips.

Contents

Chapter 1: Introduction

Welcome!

This guide describes how to install, configure, and develop applications for the webMethods SAP R/3 Adapter. It contains information for administrators who manage the system, and for application developers who create applications that use the system.

To use this guide effectively, you should:

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

Related Documentation

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

Refer to this book...	For...
webMethods Integration Server Administrator's Guide	Information about using the Server Administrator to configure, monitor, and control webMethods Integration Server. This book is for server administrators.
webMethods Integration Server Clustering Guide	Information about installing and configuring the webMethods Integration Server clustering feature. It also contains information for administrators who configure and manage an Integration Server system and for application developers who want to create SAP adapter services that interact directly with the Integration Server Cluster Store.
webMethods Developer User's Guide	Information about creating and testing SAP adapter services and client applications. This book is for application developers.
webMethods Integration Server Built-In Services	This guide describes the built-in services provided with a standard installation of webMethods Integration Server and located in the WmPublic or WmDB packages.
Serialization of ABAP data in XML	In depth information about how ABAP data is serialized in XML messages. This serialization is used for RFC and BAPI parameters. You will find this document at: *Integration Server_directory*\packages\SAP\doc\ABAPSerialization.html

Refer to this book...	For...
RFC-XML Format Specifications	In depth information about how RFCs, BAPIs and IDocs are formatted in XML. You will find this guide at: *Integration Server_directory*/packages/SAP/ doc/IFR-XMLFormatSpec.pdf
RFC-XML Specification	In depth information about the RFC-XML specifications. You will find this document at: *Integration Server_directory* /packages/SAP/ doc/RFC_XML.html
BizTalk Envelope Specifications	In depth information about how the SAP adapter uses the BizTalk Envelope for the transmission of business documents. You will find this guide at: *Integration Server_directory* /packages/SAP/ doc/BizTalkEnvelopeSpec.pdf
webMethods Integration Server IDoc Class Documentation	In depth information about the IDoc Java classes. You will find this guide at: *Integration Server_directory* \packages\SAP\ doc\api\index.html
webMethods Integration Server API Reference	Descriptions of the Java classes and built-in services you use to create SAP adapter services. You will find this book at: *Integration Server_directory* \doc\API\index.html
Microsoft BizTalk Framework 1.0a *Independent Document Specification*	This add-on package introduces XML messages that are based on the Microsoft Biztalk Framework. Information on this framework and the XML envelope it defines can be found on the Web site http://www.biztalk.org.

Chapter 2: Product Overview

What is the webMethods SAP R/3 Adapter?

The webMethods SAP R/3 Adapter (SAP adapter) allows you to extend your business processes over the Internet and integrate non-SAP products using open and non-proprietary technology. The SAP adapter allows for bi-directional, real-time, asynchronous communication to and from the SAP server. You can:

- Execute SAP's implementation -independent 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 SAP adapter services that execute a BAPI. Applications within your organization can then invoke the SAP adapter services to execute a BAPI on the SAP server. 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 as well as asynchronous processing can be triggered by these interfaces. Asynchronous processing uses the ALE services inside the SAP system transparently for the client.

- Execute SAP remote function calls (RFCs) from webMethods Integration Server**.** You can access all SAP functionality that is available via RFC from webMethods Integration Server. External applications do not need to understand SAP data types, ABAP structures, or the RFC protocol to communicate with an SAP system.

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

- Route SAP business documents (IDocs) based on criteria you specify. webMethods Integration Server provides rich routing capabilities for IDocs. There are a number of different transport types available out-of-the-box. These include routing of an IDoc to another SAP system, to a mySAP.com Marketplace, or simply to a remote URL in an XML format. The SAP adapter allows you to increase efficiency across the supply chain and customer loyalty by tightly integrating your business infrastructure with that of any partner. Typical deployment scenarios for SAP adapter can be:

- Real-time integration between supplier inventories and your SAP system.

- Real-time integration between product, price and availability information from any number of suppliers and your purchasing application.

- Real-time integration between fulfillment and order tracking applications and your shippers' internal systems.

Functional Highlights

- Synchronous and Asynchronous Communication with SAP Systems via RFC and tRFC

- Bidirectional communication to and from SAP systems

- Higher level services to process SAP IDocs and BAPIs

- Easy XML- and Internet- enabling of existing SAP releases
- Support of BizTalk XML envelopes for BAPI and RFC calls
- Support of unified error handling of BAPIs and RFCs on XML level

Complete SAP System Integration

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

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

As a special service, SAP IDocs can be converted to a structured document 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. On the other hand, you may simply access existing BAPIs in SAP systems from a browser or client application or send XML documents to the SAP system.

Developing applications with the SAP adapter requires no knowledge of SAP data structures, significantly reducing deployment time and cost.

Integrating SAP Systems over the Internet

Most business-to-business scenarios demand that two systems communicate with each other securely over the Internet, despite existing firewall restrictions. The SAP adapter leverages HTTP to seamlessly exchange data between two or more SAP systems across the Internet, without requiring changes to the existing security infrastructure.

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

Routing IDocs, RFCs and BAPIs

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

BAPIs provide a simple way to access SAP solutions. They can be used for both synchronous and asynchronous calls to an SAP System by using the same XML message format. Asynchronous BAPI calls are provided by using the ALE services inside the SAP System. Thus you can easily leverage ALE mechanisms without having to deal with the sometimes complex IDoc format. This works for all IDocs which 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, and distribution services.

BAPI interfaces are developed according to strict development guidelines, which means 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 server can therefore be exchanged without invalidating the (BAPI) interface used by clients.

A business management function which is implemented as a BAPI can be called both synchronously and asynchronously by using the same XML message.

Support for IDoc- and RFC-XML

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

Support of BizTalk XML envelopes for BAPI and RFC calls

As a default SAP adapter will use the standardized BizTalk XML envelope format for these XML messages.

This simplifies data exchange with other Web messaging systems.

The BizTalk XML envelope differentiates between an application-specific XML body and an XML header, which is used to exchange transport-specific information, for example for routing purposes.

The BizTalk XML envelope can be used with both BAPI and RFC XML calls and also for both synchronous and asynchronous processing.

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

To allow generic error evaluations on the client, regardless of the interface type (BAPI, RFC) used, the bXML format unifies the error handling of BAPIs and RFCs.

In this way, interface type-specific error handling concepts (BAPI return parameter and function module exceptions) are converted on an XML level into a standardized type of error representation.

Built-in BAPI Tools

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

For easy identification of the relevant application interfaces, a built-in BAPI Browser has been added. 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 the routing rules for BAPIs.

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

Message Store

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

Architecture and Components

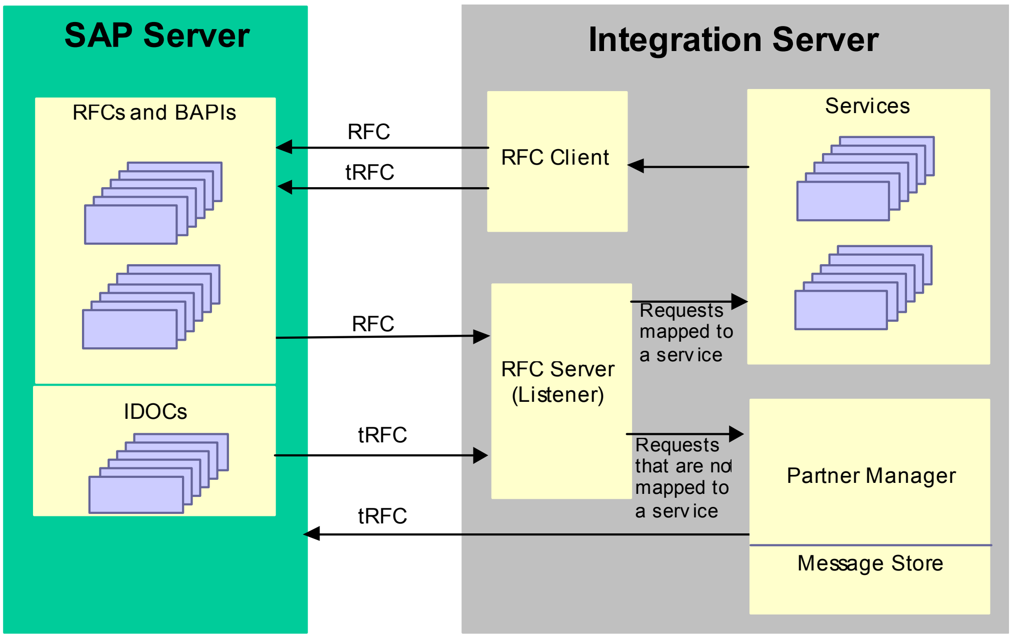
Basic Concepts

To use Integration Server successfully, you should understand the following terms and concepts:

Term	Description
BAPI (Business Application Programming Interface)	Business functions in SAP system that are written in the programming language ABAP. BAPIs are formalized RFCs. Systems that are remote to the SAP server commonly use BAPIs to have the SAP server perform an action.
RFC Server (Listener)	SAP terminology for a process that can accept RFCs from SAP systems. This allows SAP systems to access functions in external systems. In SAP ADAPTER terminology, this is a Listener. Listeners are one or more threads on the Integration Server that waits for incoming requests from SAP systems. Listeners are named and register with an SAP gateway to indicate that they are ready to accept requests. Listeners can accept RFC or tRFC requests.
RFC Client	SAP terminology for a process that sends RFCs to an SAP system to invoke functions.
tRFC (Transactional RFC)	Protocol for ensuring that an RFC is successfully executed and executed exactly once on the target system. The Integration Server can handle both inbound and outbound tRFCs.
tRFC protocol	Communications method that the SAP server uses to asynchronously invoke a function on a remote system and a remote system uses to asynchronously invoke a function on the SAP system. The transactional RFC (tRFC) protocol ensures that an RFC is successfully executed and that it is executed exactly once.
TID	Transaction ID. A 24 bytes long, globally unique identifier used by tRFC to ensure exactly one execution.
RFC (Remote Function Call)	Requests that the SAP server initiates to have functions performed on remote systems, or calls remote systems initiate to have the SAP server perform a function.
RFC protocol	Communications method that the SAP server uses to synchronously invoke a function on a remote system, and a remote system uses to synchronously invoke a function on the SAP system.
SAP ADAPTER Service	Integration Server functions that are named with a hierarchical interface/service syntax. Services can be flow (interfaces to data sources, XML documents, or HTML Web sites), Java, or C/C++ developed by you, third-party vendors, or SAP AG.
LIBRFC	Code library provided by SAP allowing third parties to integrate the RFC protocol into applications. This is how the Integration Server communicates with the SAP system.
IDoc (Interchange Document)	EDI-like SAP business document.

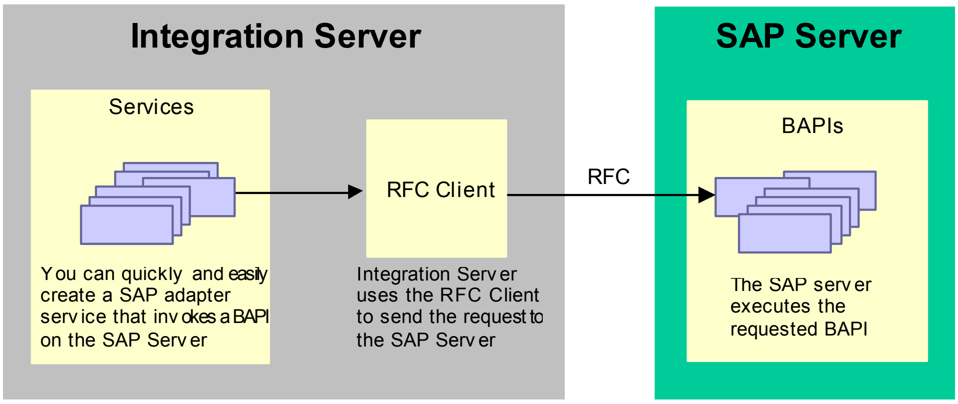
Term	Description
Partner Manager	A process that accepts messages and routes them to a configured location, for example, messages can be routed to an SAP system, an Integration Server, or a remote URL in an XML format. Typically, IDocs are routed through the Partner Manager.
Transaction and Message Store	A JDBC-compliant database that Integration Server uses to track all transactions that pass through the Partner Manager.

The following illustrates the components in a system that uses the Integration Server:

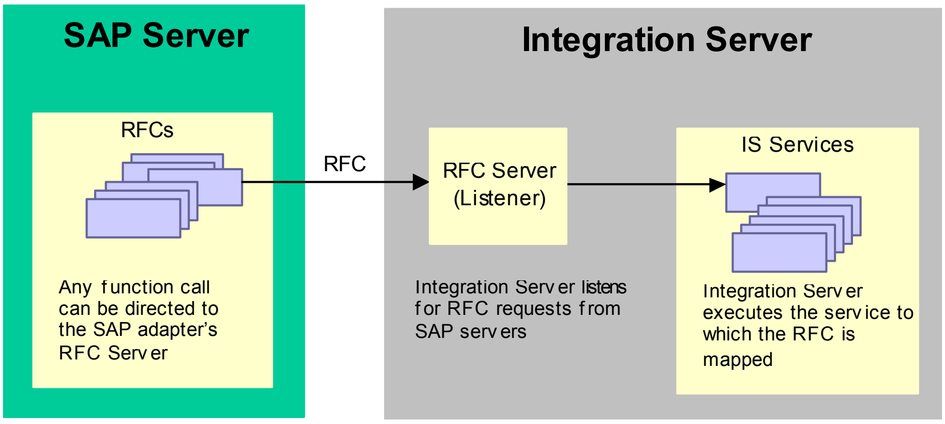


Invoking Business Logic

The Integration Server incorporates an RFC Server and RFC Client to provide real-time inbound *and* outbound communication to and from the SAP system. The server uses the RFC Client to send requests to execute BAPIs to the SAP server.


Integration Server
Services
You can quickly and easily create a SAP adapter service that invokes a BAPI on the SAP Server
RFC Client
Integration Server uses the RFC Client to send the request to the SAP Server
RFC
SAP Server
BAPIs
The SAP server executes the requested BAPI

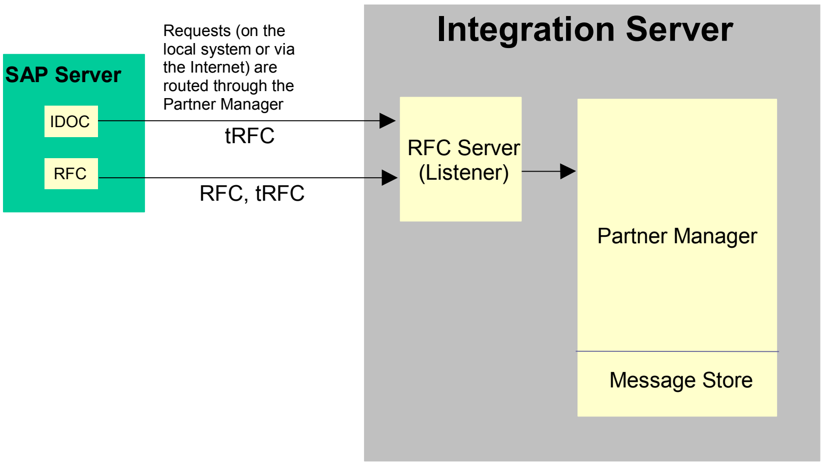
The Integration Server uses the RFC Server (Listener) to listen for incoming requests to execute SAP adapter services from an SAP server. From an SAP application point of view, calling the Integration Server is no different from calling any other RFC server.


SAP Server
RFCs
Any function call can be directed to the SAP adapter's RFC Server
RFC
Integration Server
RFC Server (Listener)
Integration Server listens for RFC requests from SAP servers
IS Services
Integration Server executes the service to which the RFC is mapped

Routing Messages through the Partner Manager

When the Integration Server receives a request on the RFC Server (Listener) that is not associated with (or mapped to) a specific service, the Integration Server sends the request to the Partner Manager. The Partner Manager can receive:

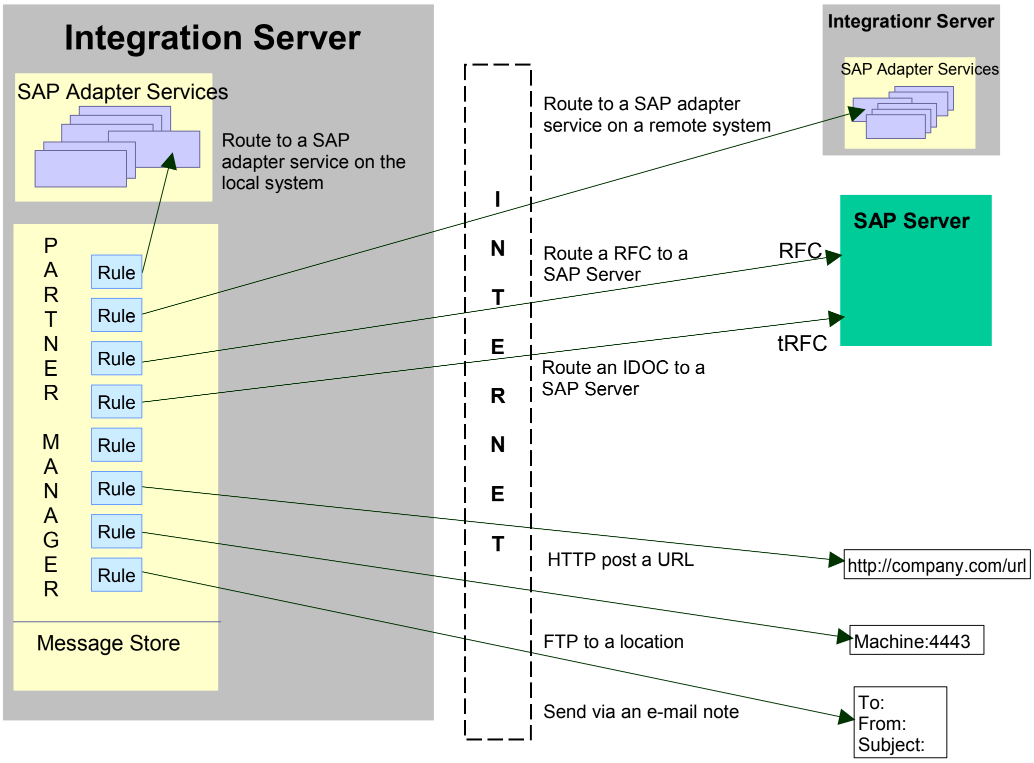
- IDocs and BAPI calls from an SAP server or IDoc-XML messages from any web client
- RFCs from an SAP server or RFC-XML and bXML messages from any web client



An Integration Server inbound call is an outbound call from the SAP system's point of view.

When the Partner Manager receives a message with tRFC or an XML-message that contains a TID, it logs the transaction in the Message Store. It then uses routing rules to determine where to route the messages. The routing rules indicate where to route the message based on who the messages is from (sender), who is to receive the message (receiver), and the message type. A message can be:

- Routed to a SAP server (IDocs, RFCs and BAPIs)
- Routed to an Integration Server service on the local machine or a remote machine
- HTTP posted to a URL
- FTP'd to a location
- Sent via an e-mail note


Integration Server
SAP Adapter Services
Route to a SAP adapter service on the local system
PARTNER MANAGER
Rule
Rule
Rule
Rule
Rule
Rule
Rule
Rule
Message Store
INTERNET
Route to a SAP adapter service on a remote system
Integrationr Server
SAP Adapter Services
Route a RFC to a SAP Server
RFC
SAP Server
tRFC
Route an IDOC to a SAP Server
HTTP post a URL
http://company.com/url
FTP to a location
Machine:4443
Send via an e-mail note
To:
From:
Subject:

Chapter 3: Clustering with the SAP Adapter

Overview

Clustering is an advanced feature to use with the SAP adapter. In particular cases, clustering may extend the reliability and availability of the Integration Server.

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

The typical goals customers seek to fulfill with clustering are high availability, scalability, and failover mechanism.

Features of webMethods clustering are:

- Inherent, customizable load balancing
- Session persistence in a central cluster repository
- A framework for check-points to save and read out the entire pipeline of a service
- Guaranteed delivery possible with B2B clients and active load balancing (even with external load balancers and clustering)
- Central monitoring of ports, packages, workload of the servers of a cluster

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

Restrictions and Prerequisites

High Availability

- The client has to use the proprietary wM Context Classes to allow server switches
- The reposerver process is a single point of failure if not installed redundantly

Scalability

- The client has to support HTTP redirect
- Scalability of the cluster is limited as load balancing itself creates overhead. The performance deteriorates with more than two servers in a cluster.

Failover mechanisms

- The client has to support cookies for session identification
- Checkpoint support requires manual coding and retry logic. Resending the entire message might be cheaper.

Protocol dependency

- Clustering is only usable when the server is called via HTTP. Login via ftp is not clusterable.

Additional Restrictions in an SAP Environment

Regarding the SAP Transaction management, please observe the following additional restrictions:Whenever it is necessary that the user context is maintained 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: an RFC-connection cannot be put into the repository server but is bound to a single server.

The method to bind a server session to an RFC connection is to embed subsequent messages of a transaction (e.g. a change BAPI and a BAPI_TRANSACTION_COMMIT) in a Lock/Unlock_Transaction service. These services are clusterable.

Evaluation Results

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

Advantages:

- No additional load balancing component is needed.

Disadvantages:

- High Availability for various kinds of HTTP clients is easier to achieve with an external load balancer and several instances of independent Integration Servers.

- Scalability is through load balancing is not reached. Independent Integration Servers fronted by an external load balancer offer a better system performance.
- Failover mechanisms only make sense for multi-step transactions. With an SAP backend system these transactions are bound to a single server anyway. So the checkpoint support of the cluster does not apply.

If you still want to apply clustering in your system environment please follow the instructions further on in this chapter.

Supporting Clustering with Integration Server

For an Integration Server cluster to run successfully with the SAP adapter, each clustered server must access the same configuration information.

With the SAP adapter, you should be aware of the following limitations and behavior in a clustered environment.

Load Balancing

This feature accommodates HTTP and HTTPS requests as well as requests from applications built using the adapter's client API.

By default, the SAP adapter takes advantage of load balancing for all client requests, including outbound RFC calls. Each host in the cluster maintains its own structure, function cache, and connection pool, which are populated on demand.

Failover Support

This feature accommodates requests from applications built on SAP's client API only.

Implementing Clustering with the SAP Adapter

To implement clustering with the SAP adapter, follow these steps. These steps assume that you have already set up your Integration Server clustered server environment using the procedures in the *webMethods Integration Server Clustering Guide*.

Step	Description
1	Prepare the clustered environment. This step involves disabling existing packages and load balancing and identifying the server that will handle administrative tasks.
2	Install and configure the R/3 IM clustered environment. This step involves installing and configuring the IM on a single administrative server and replicating the package and all required elements to each machine in the cluster.
3	Maintain load balancing. This step involves understanding how load balancing works with inbound RFCs and how to prevent request redirection for certain transactions.

Disabling Existing Packages

If any of the following packages already exist on any machine in the cluster (other than the administrative machine on which you are doing development), you must disable the package before proceeding. This ensures that there are no file conflicts during installation and setup of the clustered environment.

- SAP—This package contains all of the SAP functionality.
- WmPartners—This package contains services that provide document routing functionality.
- Other packages in which you have routing rules and mappings—If you saved routing rules or mappings to any other packages, disable them on the clustered machines. In other words, if you have any other packages that you plan to replicate later to the clustered machines, disable those packages.

➡ To disable the SAP package, it is necessary to restart the Integration Server.

For instructions on disabling packages, see the *webMethods Integration Server Administrator's Guide*.

Disabling Load Balancing

Before you begin installing and configuring the SAP adapter, make sure that you disable load balancing on each machine in the cluster. This ensures that all of the administrative services (such as creating RFC mappings) are executed on the same host during development and installation.

For instructions on disabling load balancing, see the *webMethods Integration Server Clustering Guide.*

Identifying the Administrative Host

Finally, identify the machine in the cluster that will perform all of the administrative services (such as creating RFC mappings) for the clustered environment. You will be doing the majority of development on this machine.

Procedure

To install and set up Integration Server in a clustered environment

1. Perform the steps Disabling Existing Packages, Disabling Load Balancing and Identifying the Administrative Host on page 3-6.
2. On the machine that will host all of the administrative services, make sure that the package is loaded and activated, as should be the case with any SAP adapter installation.
3. Configure the administrative machine of the SAP System by creating all required aliases, inbound and outbound mappings, routing rules, transport configurations, etc.
4. Test the mappings, routing rules, etc. for reliability and accuracy.
5. Publish each package that you want to distribute to each server in the cluster. (For instructions, see the *webMethods Integration Server Administrator's Guide*.)

Any packages that you modified on the administrative machine should be published. For guidelines, see *Which Packages Do I Publish and Replicate?* on page 3-7.

6. Replicate the published packages to each server in the clustered environment. (For instructions, see the *webMethods Integration Server Administrator's Guide*.)

   For guidelines on the packages to replicate, see "Which Packages Do I Publish and Replicate?" on page 3-7. Keep in mind that when a package is replicated, the existing package on the target machine is removed.

7. After you have replicated all affected packages and completed development, you can enable load balancing on all servers in the cluster.

8. Restart all servers in the cluster that contain the replicated packages.

Adding or Changing Administrative Services

When creating new services or updating existing services, we recommend that you always use the same machine and then redistribute the package from that machine. You should follow the steps *Disabling Existing Packages*, *Disabling Load Balancing* and *Identifying the Administrative Host* on page 3-6 to prevent file conflicts.

For details on redistributing the package or packages containing new or updated services, see the *webMethods Integration Server Administrator's Guide*.

Which Packages Do I Publish and Replicate?

Use the following table as a quick reference when you are installing or reinstalling the Integration Server clustered environment.

If you want to distribute…	Publish and replicate this package to all clustered servers…
SAP system aliases and listeners	SAP
Routing rules and mappings	The package(s) in which you saved the rules (saved as flow services) and mappings + WmPartners

Keep in mind that if you have services that reference other services in other packages, you need to replicate those dependent packages as well.

How Inbound RFCs Are Balanced

SAP listeners in the Integration Server implement self-registering RFC servers. Since 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 if one of the listeners is not busy, then no requests from the SAP System will block.

Preventing Load Balancing for Specific SAP Transactions

To execute transactions with several dependent calls in a row, you must ensure that the same SAP transactions context is used for all calls. However, with load balancing enabled, there is no way to guarantee that a request will be handled by the same server as the previous request. Therefore, you must temporarily prevent redirection of your request by calling a special service and locking your session to your connection to the SAP System.

To prevent load balancing for services called from an Integration Server session , call the pub.sap.client:lockSession service before any series of calls that must have the same server host. Provide the value of "false" for the input variable redir. This disables redirection of requests and locks your session to your connection to the SAP System. For details on this see service pub.sap.client:lockSession.

To re-enable load balancing for services called from an Integration Server session, make sure that the transaction is complete (i.e., all services have been called). Then call the pub.sap.client:releaseSession service. This unlocks your session to your connection with the SAP System. For details on this see service pub.sap.client:releaseSession.

For more detailed information on Integration Server clustering, see the *webMethods Integration Server Clustering Guide*.

Chapter 4: Calling a SAP System from a Service

Overview

You can create Integration Server services that execute a Function Module (FM) residing on an SAP server. 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 (i.e. ‘remote enabled’). This includes all BAPIs, which are formalized FMs. To create such a service, you simply select the SAP server and the RFC that you want the service to execute. After creating the service, you can create a client that invokes the service.

Before you can create a service, you must configure the Integration Server to define the SAP servers you want to use. You identify an alias name for the SAP server and specify information that Integration Server requires to connect to the SAP server.

This chapter describes how to map a service to a FM and how to test the service.

You must have administrator privileges on the Integration Server to execute these procedures. If you do not have such privileges, have your Integration Server administrator perform the following for you.

General Settings

For the general settings of Integration Server you can use the Server Administrator.

Setting	Description
Timeout (minutes)	Delay (minutes) until an unused connection to a SAP System is timed out (default: 5)
Timeout check period (seconds)	Time interval in seconds between checks whether unused pooled connections have timed out
Sessionpool size	Maximal number of connections in one RFC connection pool to one SAP System (default: 10)
Poolqueue waiting time (seconds)	Delay (seconds) until requests waiting for a 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: true and false (default)

Defining SAP Servers

Use the following procedure to define the parameters that the Integration Server will use to establish a connection to a SAP server. Integration Server requires a connection to the SAP server whenever functionality from the SAP System is to be invoked, that is whenever SAP adapter acts as a client for a SAP server. But also when SAP adapter receives a call from a SAP Server and it needs to make a callback to the calling system to look up the function interface or IDoc definition from the SAP DDIC.

To define the connection to the SAP system

1. Start Integration Server.

2. Choose **SAP** from the Administrator UI navigation panel.

3. Select the **SAP Servers** tab.

4. Choose **Add Server**.

5. Complete the following fields with the values from a server configured in the SAP GUI.

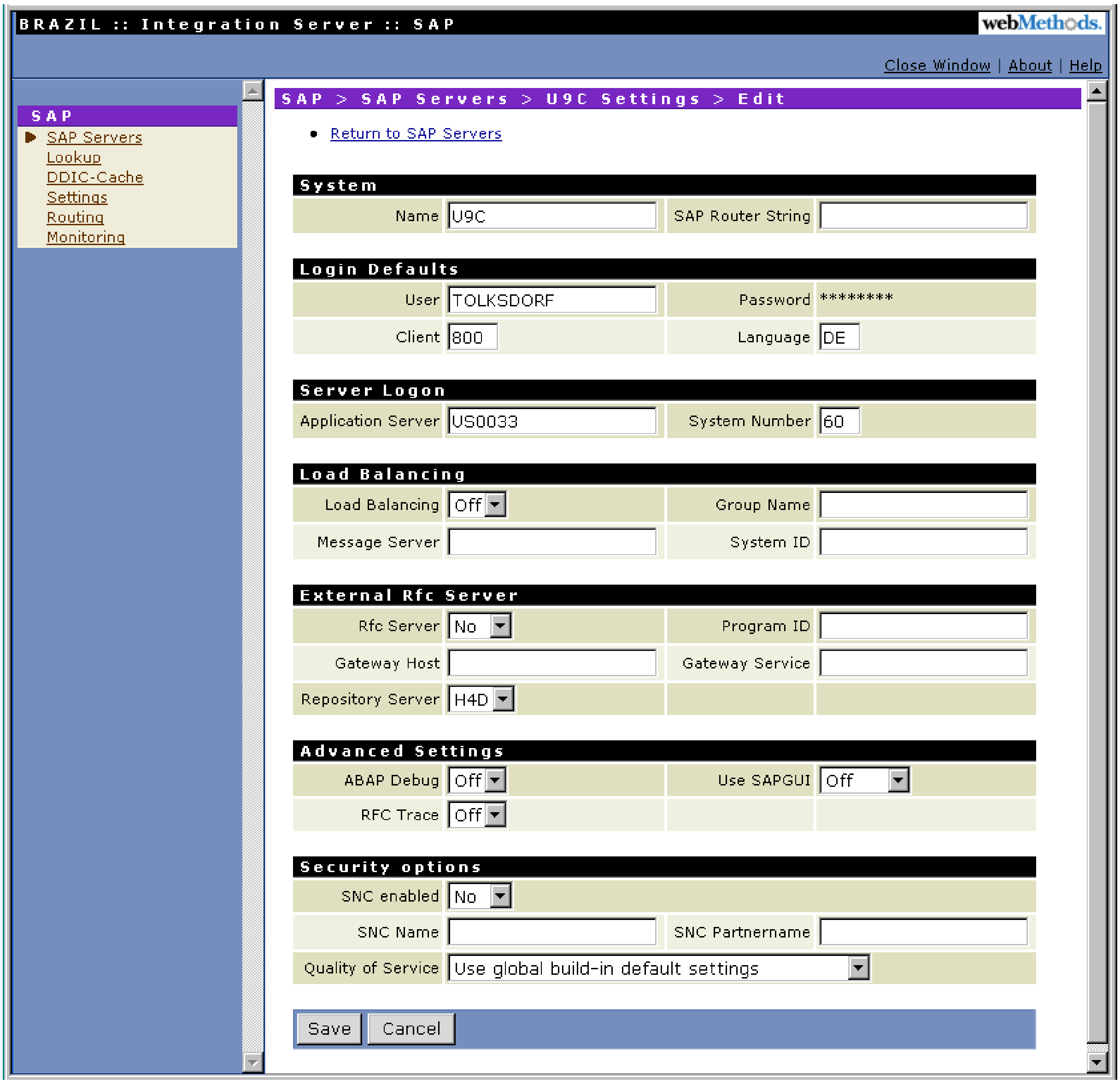
   System

In the field…	Specify…
Name	An alias for the SAP server. This is the name by which the server will be known to Integration Server developers, clients, and partners.
SAP Router String	The SAP router string. A router string contains a substring for each SAP Router to set up a connection in the route: the host name, the port name, and the password, if one was given. Example: /H/127.0.0.1/H/ Syntax: /H/ indicates the host name. The SAP router string is only needed if there is a firewall between the SAP server and the Integration Server. For more Information on configuring the SAP router see *SAP Library.*

   Login Defaults

In the field…	Specify…
User	The SAP user name.
Password	The SAP password.
Client	The three-digit SAP client number.
Language	The SAP language code. If connecting to a V3 system, this is one character. For V4, it is two characters.

SAP Server Definition


BRAZIL :: Integration Server :: SAP
webMethods.
Close Window | About | Help
SAP
SAP Servers
Lookup
DDIC-Cache
Settings
Routing
Monitoring
SAP > SAP Servers > U9C Settings > Edit
Return to SAP Servers
Save
Cancel

System

Name	U9C	SAP Router String	

Login Defaults

User	TOLKSDORF	Password	********
Client	800	Language	DE

Server Logon

Application Server	US0033	System Number	60

Load Balancing

Load Balancing	Off	Group Name	
Message Server		System ID	

External Rfc Server

Rfc Server	No	Program ID	
Gateway Host		Gateway Service	
Repository Server	H4D		

Advanced Settings

ABAP Debug	Off	Use SAPGUI	Off
RFC Trace	Off		

Security options

SNC enabled	No		
SNC Name		SNC Partnername	
Quality of Service	Use global build-in default settings		

Server Logon

In the field…	Specify…
Application Server	The IP address or host name of the SAP system. Usage: `<IP>/S/3370` Syntax: /S/ is used for specifying the service (port); it is an optional entry, the default value is 3299.
System Number	The SAP system number (0-99).

You can leave all remaining fields at their default values. If you do so, you always connect to the same application server.

Load Balancing

Complete the following fields only if you apply the group login concept, (in this case you do not need to enter values in the 'Server Logon' section):

In the field…	Specify…
Load Balancing	If you choose ON, the group login concept is active.
Group Name	The name of the group you want to login.
Message Server	The host name or IP address of the message server.
System ID	The 3 digit name of the SAP system.

External RFC Server

Complete the following fields if you are using an external RFC Server. The 'Login Defaults' settings should then point either to the Gateway, at which this RFC Server is registered, or to the 'Repository Server' selected below. The 'Server Logon' and 'Load Balancing' settings can be ignored in this case.

In the field…	Specify…
RFC Server	If you choose ON, an external RFC Server will be used.
Program ID	The Program ID of the external RFC Server
Gateway Host	Gateway Host for accessing your RFC Server. Must be the IP- or DN-address of the Gateway the RFC Server is registered on.
Repository Server	This server is taken from the existing SAP server list. All repository lookups such as metadata lookups for structures and function interfaces are done there. This helps using third-party RFC Servers for the adapter as client. They can provide their functionality without being extended by a specific interface, for example, RFC_GET_FUNCTION_INTERFACE
Gateway Service	This corresponds to the system number of your RFC Server. Syntax: The Gateway Service must in the form of sapgwXX, where XX is a value from 00 to 99. Example: sapgw07

Any server program that is enabled for RFC communication can be an external RFC Server.

In order to use an external RFC Server with SAP adapter, the RFC Server must implement at least the following function modules:

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

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

Advanced Settings

In the field…	Specify…
ABAP Debug	`0` / `1`: without/with ABAP debugger (Default: 0)
Use SAPGUI	Run RFC with SAP GUI, `0` / `1` / `2`: without/with/invisible SAP GUI between two RFC functions (Default: 0)
RFC Trace	Enables the creation of RFC trace information (`0`/`1`: without/with trace, Default: 0) For detailed information on using the RFC trace see pages 5-5 and 5-6

Security Options

In the field…	Specify…
SNC enabled	Whether this server should use SNC or not. Default: no
SNC Name	Your own SNC name if you don't want to use the default SNC name
SNC Partnername	SNC name of the SNC partner (RFC server) or SNC name of the message server (Load Balancing)
Quality of Service	SNC Quality of service, possible values: • Plain text, but authorization • Each data packet will be integrity protected • Each data packet will be privacy protected • Use global build-in default settings • Use maximum available security

6. Choose **Save** to commit these changes.

If you want to use SNC connections for the communication, there are some more general and user specific settings required on the R/3 application server. These settings are listed below:

R/3 Settings for SNC	Table/Parameter
General R/3 settings for SNC:	Table SNCSYSACL (SM30, view VSNCSYSACL, TYP=E) SNC name entry of the SAP Buisness Connector
Settings for Certificate log in:	
General Settings:	Profile Parameter: SNC/libsapsecu=./libsecude.sl Snc/extid_login_diag=1 Snc/extid_login_rfc=1
	Table SNCSYSACL (SM30, view VSNCSYSACL, TYP=E), activate: • Certificate log in • Diag • RFC
User specific settings:	Table USREXTID, Category DN Entry with DN from user certificate The Seq.No. 000 is the default entry. Set entry ‘active’

For detailed information on the SAP application server settings for SNC see the corresponding R/3 documentation.

Starting with Integration Server 4.0.1, the password can no longer be changed in the former Edit screen. There is now a separate screen, on which the password can be changed. To display this screen, choose **SAP** -> **SAP Servers** selecting the Alias of your Server and then click on the "Change password" link.

Testing the Connection to SAP Servers

Use the following procedure to verify that the Integration Server can successfully connect to an SAP server that you have defined.

To test the connection to an SAP server

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **SAP Servers** tab if it is not already selected.

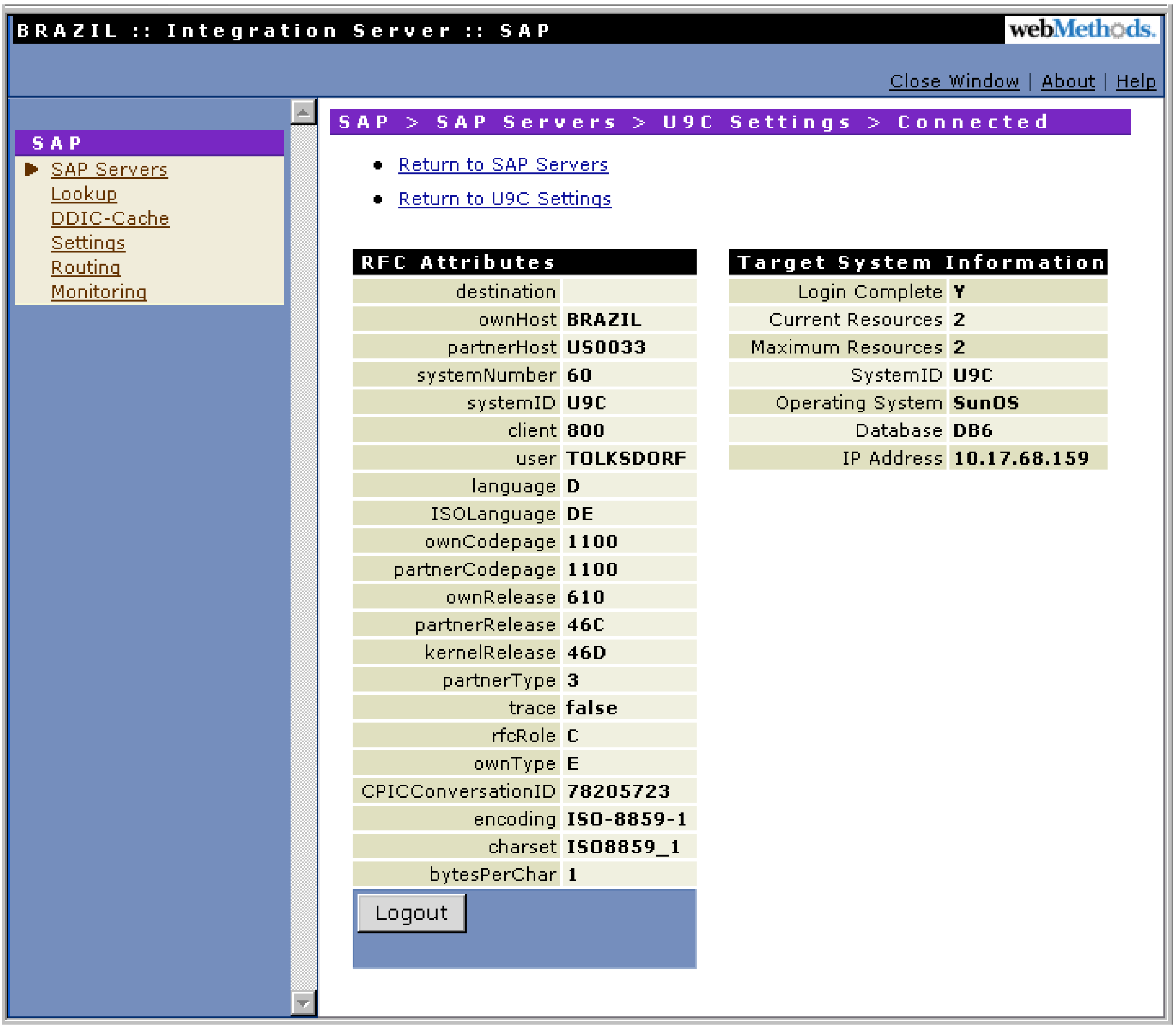
3. Press the name of the SAP server for which you want to test the connection.

4. Choose **Test Connection**.

- If the Integration Server can successfully connect to the specified SAP Server, it will display connection information as shown below.

- If you receive an error message, choose the *SAP Servers* tab and verify that your server configuration information is correct. See previous procedure.

- In case of an RFC error message, an RFC trace file named dev_rfc.trc will be written to the server directory. This file contains a more detailed description of the error cause.



RFC Attributes	
destination	
ownHost	BRAZIL
partnerHost	US0033
systemNumber	60
systemID	U9C
client	800
user	TOLKSDORF
language	D
ISOLanguage	DE
ownCodepage	1100
partnerCodepage	1100
ownRelease	610
partnerRelease	46C
kernelRelease	46D
partnerType	3
trace	false
rfcRole	C
ownType	E
CPICConversationID	78205723
encoding	ISO-8859-1
charset	ISO8859_1
bytesPerChar	1

Target System Information	
Login Complete	Y
Current Resources	2
Maximum Resources	2
SystemID	U9C
Operating System	SunOS
Database	DB6
IP Address	10.17.68.159

Testing that the SAP Server Will Execute an RFC

Once you have verified that you can connect to an SAP Server, you can use the following procedure to verify that it will accept and process an RFC request from the Integration Server.

To test an RFC from the SAP Adapter

1. Choose **SAP** from the Administrator UI navigation panel to display the main SAP page if it is not already displayed.

2. Select the **Lookup** tab.

3. From the drop down list in the **Server Name** field, select the name of the SAP server on which the FM you are to test resides.

4. Enter the name of the FM you want to test in the field Function Name of the group Function Search. For example, enter RFC_FUNCTION_*. Then select the Search button. The SAP adapter searches for all RFCs that begin with RFC_FUNCTION_.

5. In the list of matching RFCs that is returned, follow the link of the RFC you want to test. For this example, choose **RFC_FUNCTION_SEARCH**. The SAP adapter displays the function signature for the selected FM as shown below. It lists the imports, exports and tables comprising this FM.

6. Choose **Test Function** to invoke the FM on the SAP server you selected in step.

7. Enter RFC_* in the field **FUNCNAME** and press **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.

Creating a Service that Executes an RFC

To create a service that executes an RFC, you create an outbound map for the FM. This feature allows FM to be exposed as standard Integration Server services. After you create the Integration Server services, business partners, who are interested in calling these services, can download specifications for the services and generate client code.

Calling a BAPI according to its definition in the BOR is currently not possible with an outbound map. In this case you have to use the BAPI transport.

An Integration Server outbound call is an inbound call from the SAP system's point of view.

Terminology

To use Integration Server 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.

Term	Description
Function Signature	Specifications of the import, export, and table parameters of an FM. Also known as a function interface.
Import	Input parameter to an RFC. Input parameters are simple values (for example, string or number) or structures.
Map	SAP adapter terminology for an association between an FM on an SAP system and a service on Integration Server. These associations can be created for inbound (SAP calling SAP adapter) and outbound (SAP adapter calling SAP) RFCs. Maps can also be used to rename parameters or filter out selected parameters entirely.
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 FM can modify these tables and return them. Tables, themselves, are no different from relational database tables, consisting of a series of fields and rows of data for these fields.

Creating an Outbound Map

Use the following procedure to create an Integration Server service that makes a remote function call to an FM on the SAP server.

This example shows how to create an outbound map for RFC_FUNCTION_SEARCH, then test the map using SAP adapter.

An Integration Server outbound call is an inbound call from the SAP system’s point of view.

To create an outbound map for an RFC

1. Choose **SAP** from the Integration Server navigation panel.

2. Select the **Lookup** tab.

3. From the drop down list in the **Server Name** field, select the name of the SAP server on which the FM for which you want to create an outbound map resides.

4. In the **Function Search** section of the screen, type all or part of the name of the FM for which you want to create an outbound map. Use pattern-matching characters if you are unsure of the complete name and want the SAP adapter to search for several RFCs with similar names. For this example, enter RFC_FUNCTION_* in the **Function Name** field.

5. Choose **Search**. Integration Server displays a list of RFCs that match the criteria you specified in the previous step.

6. Follow the link of the name of the RFC for which you want to create an outbound map. For this example, RFC_FUNCTION_SEARCH.

7. On the next screen, follow the link **Define New Map** corresponding to an outbound request (SAP adapter calling SAP).

8. Set the parameters on the Outbound Map for <FM_Name> screen as follows. (Leave all other fields at their default values.)

In the field…	Specify…	Description
Folder	Demo	This is the name of the folder in which the service you are mapping resides.
Service	RFCFunctionSearch	This is the name of the service to which you are mapping the FM.
Package	Default	This is the name of the package in which the service you are mapping resides.
ACL	<None>	In this field you can select an ACL (Access Control List), if you want to restrict the use of this service to users allowed by this ACL.

⚠ Integration Server folder and service names are case sensitive, as are the parameters to the flow service.

Choose **Save** to add this map to the Integration Server.

Searching for Maps without Reference

For outbound and inbound Maps associated with an SAP server alias, there is since release 4.6 an option to search for maps without reference in the map overview. This can occur when you distribute a package containing inbound/outbound maps from a development system to a production system. In the production adapter the maps exist, but are not referenced in the function map list of the associated server.

To add maps without reference

1. Select **SAP** -> **SAP servers** in the Administrator UI.

2. Click on the Maps icon of the server to be configured.

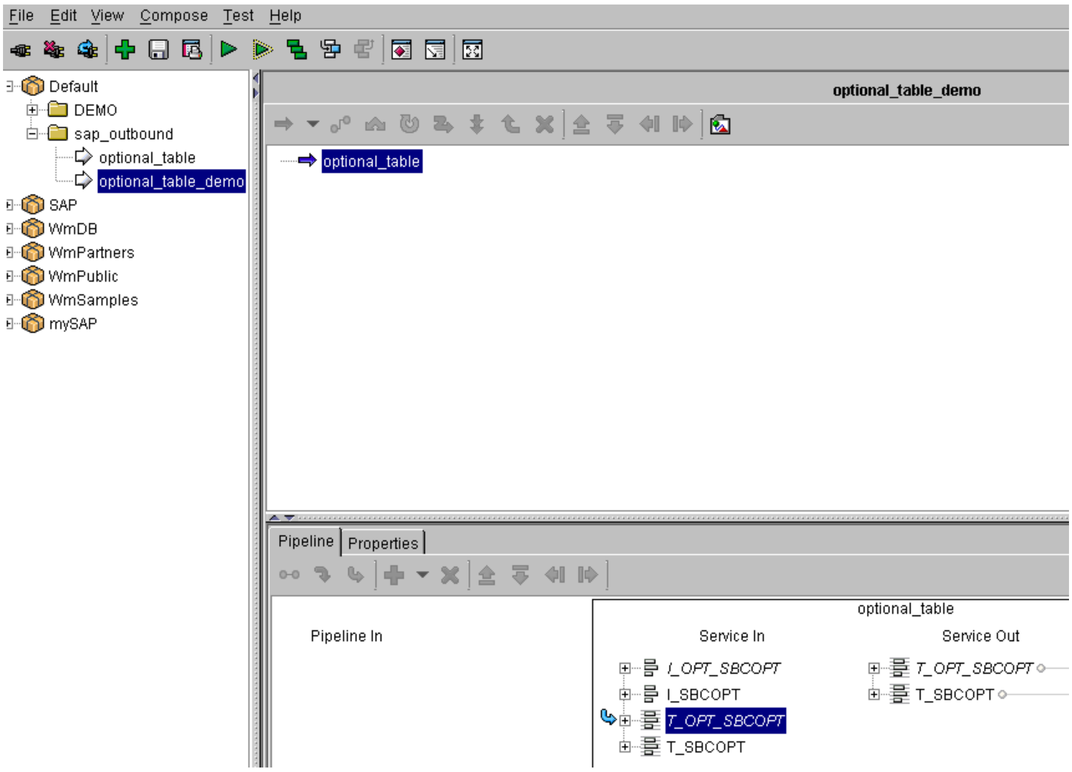
3. Click on the **Find maps for <serverName>** link at the top of the 'RFC function maps' screen. You can view then a list with all maps found.

4. Select the maps you want to add and press the **Add** button. The corresponding maps will be added to the function map list and will be referenced from now on.

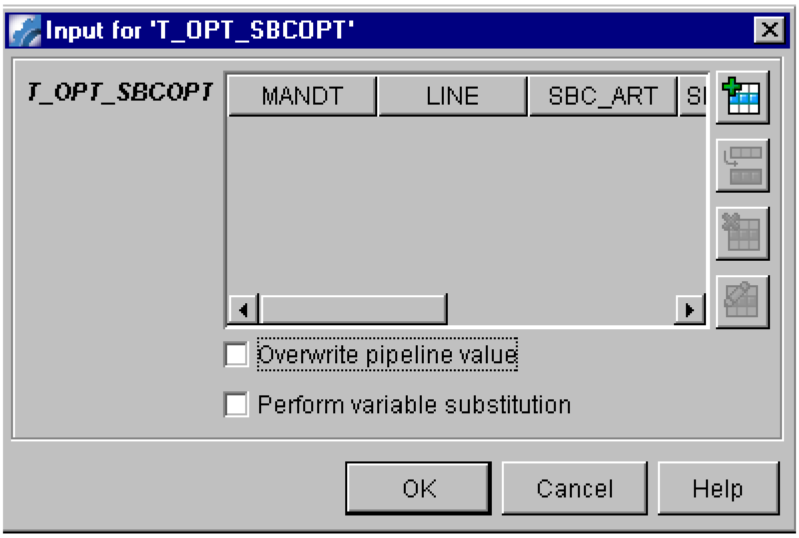
➡ There can be a single reference for an outbound map in the map list. If there are multiple entries for outbound maps of the same function module, only the first one will be referenced. Although all of them will work, it is still preferable to have a single outbound map that is called from all locations.

Using Optional Tables in Outbound Maps

Since release 3.5.2 optional tables are really optional, i.e. if an optional table is not passed to an outbound map there won't be any values returned for that table. If this is optional, then how can the outbound map return one specific table? The answer is simple:



Example: you are calling an outbound map named optional_table_demo. In this outbound map there is an optional table called T_OPT_SBCOPT. To pass this table use the "Set value” button for this parameter.



T_OPT_SBCOPT

MANDT	LINE	SBC_ART	SI

□ Overwrite pipeline value

□ Perform variable substitution

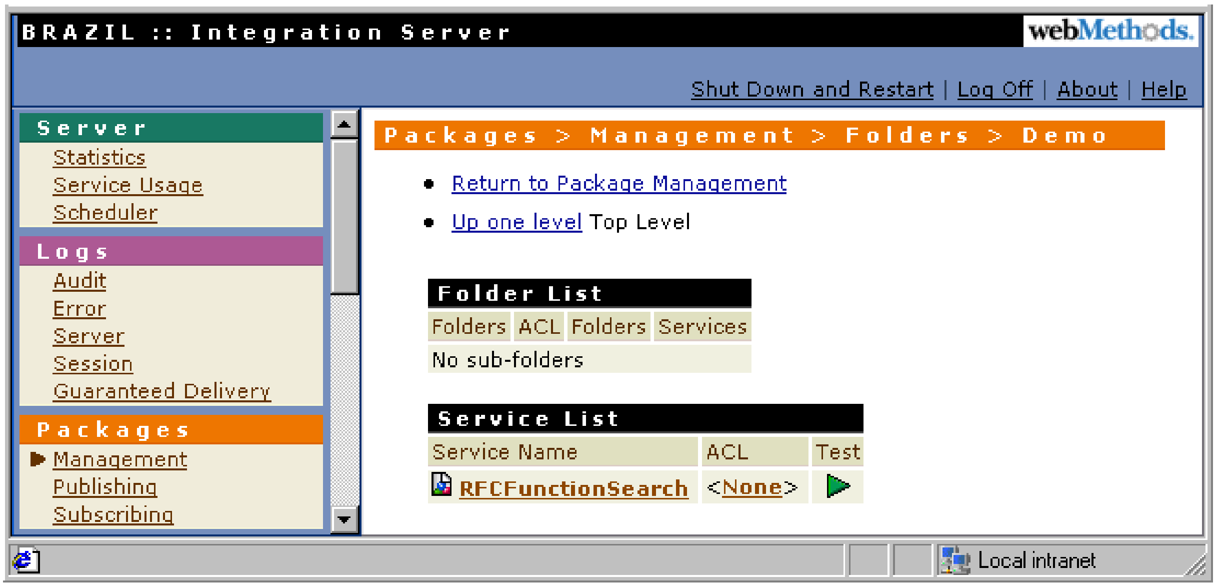
The only thing you have to do now is to uncheck "Overwrite Pipeline Value" if the empty table should only be the default value. With these settings the optional table will always be returned, both if a filled table is passed and if there is none.

Testing the Function Module

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

1. Choose **Packages** -> **Management** -> **Browse Folders** from the Integration Server navigation panel. A list of available top level folders appears.

2. Follow the link **Demo** to view the services within the BAPI interface.
You will see your service RFCFunctionSearch. This is the service mapped to RFC_FUNCTION_SEARCH.



Packages > Management > Folders > Demo

- Return to Package Management
- Up one level Top Level

Folder List

Folders	ACL	Folders	Services
No sub-folders			

Service List

Service Name	ACL	Test
RFCFunctionSearch	<None>	▶

3. Choose the checkmark **Test** that corresponds to the RFCFucntionSearch service.
   The **Test 'Demo:RFCFucntionSearch'** screen appears. This screen contains fields you use to specify input into the service.

4. Enter RFC_* into field **FUNCNAME** to specify input.

5. Choose **Test with inputs**.
   The results of RFC_FUNCTION_SEARCH will be displayed in table form in the browser.

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

Testing that the SAP Server will Execute an RFC-XML

You can invoke a function module with RFC-XML from the Integration Server via the *Lookup* screen of the Server Administrator user interface. 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 Integration Server

1. Choose **SAP** from the Integration Server navigation panel.

2. Select the **Lookup** tab.

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

4. Choose **RFC-XML**. This generates a form containing the XML template necessary to invoke the function module. Fill in the appropriate inputs and choose **Invoke on <SAP server>**. 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).

Posting an RFC-XML Document from an HTTP Client

There are several possible ways to invoke the outbound map which will call a function module. It is important that you understand the namespace convention for services (see *Administration Guide 4.6*). To invoke the service illustrated previously you need to call the URL http://*Integration Server_directory*:*port*/invoke/*folder1.folder2*/*service.*

In an interactive scenario you can simply call it from a browser the same way you call any other service on the Integration Server. The input parameters must then be URL encoded, for example:

http://localhost:5555/invoke/demo/rfcGetStructureDefinition?TABNAME=USER01

The response can then be rendered via an HTML template (for more information please see the *webMethods Templates and DSPs Guide*).

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 FM can be generated from the Lookup menu by clicking the 'RFC-XML' button. To invoke the outbound map with it you have to post it in the HTTP body of the request to the URL (as above). Your application also has to set some keys in the HTTP Header:

```
POST /invoke/<your complete service name> HTTP/1.1
--> the service name to be called
Host: <xyz>
--> your Integration Server host
Content-type: application/x-sap.rfc
--> you have to set the content type. When sending an RFC-XML document, the content
type should be application/x-sap.rfc
Cookie: ssnid=<4711>
--> This is optional. On the first connect Integration Server will generate a
session cookie. When you use this for the following communication you will be
assigned to the same Integration Server session. If you do not use the session
cookie a new Integration Server session will be created with every HTTP request.
Content-Length: <the length of the HTTP Body>
```

The HTTP Body then has to consist of the XML document, encoded in RFC-XML. Depending on whether the call was processed successfully or not you will get a response or an exception XML document (see the document <RFC-XML>). You can also force the SAP adapter to invoke the Function Module with tRFC by setting a transaction ID in the Envelope of the XML document. However, you should bear in mind that only FMs without return parameters can be called with tRFC.

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

1. Using Developer, navigate to the service pub.client:http.
2. Choose **Test → Run**.
3. Specify the URL of the RFC-XML handler service. For example, http://localhost:5555/invoke/pub.sap.transport.RFC /InboundProcess (if the document contains routing informations).
4. Specify Post as the method.
5. Add one entry in the field **headers**:

```
Name: Content-type
Value: application/x-sap.rfc
```

Creating a Service that Executes a BAPI

The lookup-tool has been enhanced by adding 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 **Server Name**.

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 pressing on *bXML*.

- In the group **BAPI By Name** enter only the name of the business object. Choose *Lookup* to view details of the business object and select one BAPI from the list of available BAPIs for the business object.

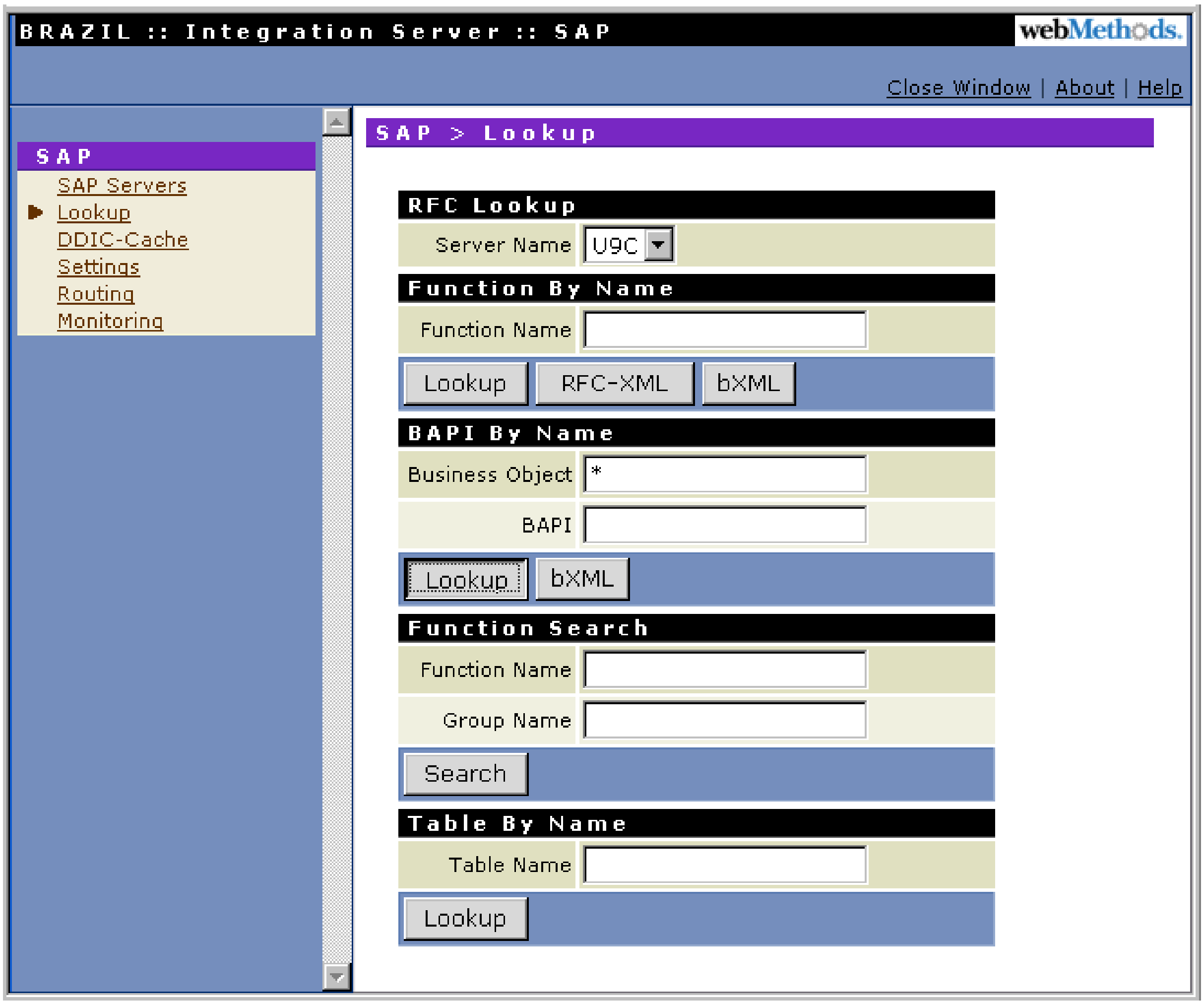
- In the group **BAPI By Name** enter * in the business object field to view a list of all business objects available in the selected system.

Business object 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 in 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. Please note that this list may also contain business objects that do not contain BAPI methods.



Server Name U9C

Business Object *

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 detail 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
Business Object	The name of the corresponding business object is listed with a link to the overview page of this business object.
BAPI	The name of the BAPI is displayed in the same case-sensitive style, as used with XML messages.
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.
Factory method	The Factory method flag is true, if the BAPI is used to create an object instance inside the SAP System. Factory methods return the key fields of the business object.
Implementing RFC	The Implementing RFC is the message type that has to be used to set up routing for synchronous calls coming from a 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 using 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 Integration Server 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
Business Object	The name of the business object and BAPI to which this parameter belongs are displayed with links to the corresponding overview pages.
BAPI	The name of the BAPI is displayed in the same case-sensitive style, as used with XML messages.
Parameter	The name of the parameter is displayed in the same case-sensitive style, as it has to be used in an XML message.
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
Business Object	The name of the business object and BAPI to which this parameter belongs are displayed with links to the corresponding overview pages.
Key field	The name of the key field is displayed in the same case-sensitive style, as it has to be used in an XML message.
Internal name	
ABAP Dictionary Type	The ABAP Dictionary Type is a link to the data type description used for this parameter. This is usually a field of a structure inside the SAP Data Dictionary. By following the hyperlink, you will see the whole structure.

Please note that key fields are used with instance-dependent (non-static) methods and with factory methods.

In calls to these methods, the key fields of the corresponding business object have to be specified as attributes of the business document root element.

Using the key field `CompanyCodeId` in a call to the *`CompanyCode.GetDetail`*:

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
    <header>
       …..
    </header>
    <body>
     <doc:CompanyCode.GetDetail CompanyCodeId="0001"
           xmlns:doc="urn:sap-com:document:sap:business"
           xmlns="" >
      </doc:CompanyCode.GetDetail>
        </body>
  </biztalk_1>
```

Generating XML Calls for a BAPI

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

The page displays the XML message needed to call the BAPI which is wrapped in the BizTalk XML envelope. You can edit the XML and enter your specific data into the pre-generated XML elements for parameters and key fields. You can also adjust the pre-generated information in the BizTalk XML header fields that are filled with default data.

There are three ways of calling the BAPI, which can be selected through the dropdown 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 is asynchronously callable (please also consider the installation prerequisites in chapter 3).
- Applying routing rules: The call will be sent to the BAPI inbound process of the Partner Manager that will check if a routing rule for this BAPI call exists and will forward the call to the specified service. This will only work, if a correct routing rule has been setup, otherwise you will receive an XML error-message.

When you select one of the first two options, the SAP adapter will execute a synchronous or asynchronous call without checking the existence of a transaction identified in the XML message. However, when using routing or when posting directly to adapter services, you have to specify a transaction identifier in the XML message, if you want to execute an asynchronous call. If you want to execute a synchronous call, you do not need to specify a transaction identifier.

After choosing the invoke button, the result of your XML call is displayed. In 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 reference on how to call a BAPI from an external client using an XML document, see *Posting BAPI-Based XML to the Integration Server* on page 7-3.

Chapter 5: Calling Services from a SAP System

Overview

You can create RFCs on the SAP server that invoke Integration Server services. This allows SAP users to send the information that is available to the Integration Server.

Before you can create an RFC that invokes a service, you must configure the SAP server to have an RFC destination for an Integration Server. This sets up where the SAP server sends RFCs that invoke an Integration Server service. You must also configure the Integration Server to have a listener (RFC Server) that listens for RFCs from the SAP server.

After you have the SAP server and the Integration Server configured, you can create a function module on the SAP server that requests the execution of an service. You also need to create an inbound map on the Integration Server. The inbound map indicates what service the Integration Server is to execute when it receives the RFC from the SAP server.

Creating a RFC Destination on a SAP System

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

Register the Integration Server as an RFC Destination

Use the following procedure to configure the Integration Server as a registered RFC destination on the SAP system.

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 Integration Server as an RFC destination

1. Use the SAP GUI to login to the SAP system.

2. Choose **Administration** → **System Administration** → **Administration** → **Network** → **RFC Destinations (transaction SM59)**.

3. Choose TCP/IP connections.

4. Choose **Create**.

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

⚠ This field is case sensitive. We recommend that you pick a name that is all UPPERCASE characters.

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

7. Enter `Integration Server` in section **Description**.

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

9. Choose **Registration** as **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. Choose **Save** from the toolbar or select **Save** from the **Destination** menu.

12. Choose **Destination** → **Gateway options**.

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

14. Enter `sapgw<sap system number>` in field **Gateway service**. This guarantees that you can access the RFC Server from all SAP application servers.

15. Choose **OK**.

16. Choose **Save**. Remain on the current screen while you complete the steps for creating an Integration Server RFC Listener.

Configuring an RFC Listener in the Integration Server

Integration Server requires an RFC listener (RFC Server) to listen for inbound RFC requests from an SAP server.

Use the following procedure to create a listener on the Integration Server to respond to RFCs issued by the SAP server.

To create an RFC Listener on the Integration Server

1. Choose **SAP** from the Integration Server navigation panel.

2. Select the **SAP Servers** tab, if it is not already selected.

3. In the **Listeners** column list of installed SAP servers, select the number (initially *0*, indicating that there are no listeners defined) corresponding to your SAP server for which you want to create a listener.

4. Choose **Add Listener**.

5. Complete the following fields of the SAP Listener Definition page. (Leave all other fields at their default values.)

Field	Specify
Program ID	The Program ID that you specified when creating the corresponding RFC destination on the SAP server. This field is case sensitive.
Number of Threads	The number of simultaneous incoming RFCs that this Listener can handle.

Field	Specify
Gateway Host	Gateway Host for accessing your SAP server. This must be exactly the same parameter as you chose for the corresponding RFC destination in the SAP system.
Gateway Service	The Gateway Service. This corresponds to your SAP system number. If your SAP system number is "01" then your gateway service is "sapgw01." Again, this must be exactly the same parameter as you chose for the corresponding RFC destination in the SAP system.
Autostart	Whether this listener will start automatically when the Integration Server starts. Select **Yes** to have Integration Server automatically start the listener when the Server is started. If you select **No**, you will have to manually start the listener after the Integration Server is started. To manually start the listener, from the **SAP Servers** tab page, choose the number in column **Listeners** for the appropriate SAP server. Then, press the red icon in field **Started?.** The icon becomes green when the listener is started. Once a listener is enabled, a connection exists between the Integration Server and the SAP server.
RFC Trace	Whether you want RFC tracing enabled or disabled. For a productive system, select **Off**. When you select **ON**, Integration Server collects tracing in *.trc* files in the Iintegration Server installation directory. The names of the .trc files all begin with rfc. (For detailed information see *Managing RFC Trace Information* and *Viewing and deleting RFC Trace Files*, pages 5-5, 5-6).
Repository Server	The SAP server 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.

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. The following fields must be completed thoroughly and accurately. If this information is incorrect or left blank, the Listener will fail to start.

6. Choose **Save** to commit these settings to the Integration Server.

7. To start the listener, press the red ball in the **Started?** column. After a short pause, the ball turns green.

If an error occurs during startup, the state ball will turn yellow when refreshing the page. Click the yellow ball to receive the error message. Error messages at this stage typically indicate a problem with either the listener configuration or the network. Review the listener settings and check the network.

Managing RFC Trace Information

In the development and test phase it is sometimes helpful to view RFC traces, if there are problems with the function modules, you are trying to call. In order to avoid the need to access the file system of the machine, on which the Integration Server is running, it is now possible to view and delete RFC Trace Files and SAP Log Files from the Administrator UI.

Testing the RFC Listener

Use the following procedure to verify that the SAP server can successfully issue a remote function call (RFC) to the Integration Server.

To test the RFC Listener

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

    1. Choose **Administration** → **System Administration** → **Administration** → **Network** → **RFC Destinations (transaction SM59)**.

    2. Open the TCP/IP connections folder.

    3. Select the RFC destination you previously created.

2. Select the **Test Connection** toolbar button.

    If the SAP server can successfully connect to the Integration Server RFC 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.

Monitoring Information for RFC Listeners

Viewing and Deleting RFC Trace Files and SAP Log Files

In the Administrator UI browse to **SAP → Monitoring**. Here you can view and delete RFC trace files and SAP log files.

RFC Trace Files are split into several parts, each of them containing one trace entry along with the date and time it was created. Large SAP Log Files are split into several parts of about 500K. 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 Integration Server is still writing to it), you will get a notification, and the file is not deleted.

Performance Output Information in the SAP Log File

From debug level 7 or higher you can view performance output information for the Integration Server in the SAP log file. The corresponding terms are described below:

The term…	Specifies…
No. of calls	The number of calls that were traced with this Performance Object. For client performance data this is normally 1, if a session is locked or for listener performance data it can be more than one.
No. of bytes sent	Total number of (uncompressed) bytes of data sent from the adapter to the SAP system. In the RFC layer that might be less because of compression.
No. of bytes received	Total number of (uncompressed) bytes of data received by the adapter from the SAP system. In the RFC layer that might be less because of compression.
Time for marshalling (ms)	Time needed for transferring the data from JCo Java Objects to RFC C structures.
Time for unmarshalling (ms)	Time needed for transferring the data from RFC C structures to JCo Java Objects.
Time for middleware calls (ms)	Time needed for executing a function module in an SAP system.
Time for handling request (ms)	Time needed to handle an inbound request from an SAP system in the adapter after unmarshalling and before marshalling the data.
Total elapsed time (ms)	Total time needed for handling inbound/outbound calls in JCo layer. (1)-(8) is the performance data of the JCo layer. In addition to that there is data available for the adapter data layer seen in (9)-(14).
BC Data: Time for marshalling (ms)	Time needed for transferring the data from adapter Objects (pipeline representation) to JCo Objects.
Time for unmarshalling (ms)	Time needed for transferring the data from JCo Objects to adapter Objects (pipeline representation).
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.
Time for rfc calls (ms)	Time needed in the JCo layer for outbound calls. Should be equal to (8)
Time for adapter service calls (ms)	Time needed for the invocation of a service to handle an inbound request in the adapter after preparing/marshalling and before unmarshalling the data. (Listener performance data).
Total time for function calls (ms)	Total time needed for handling inbound/outbound calls in adapter layer.

jARM (Java Application Responsetime Measurement)

The purpose of this type of monitoring is to give an overview of the status of a JAVA system. You'll find this information also on the SAP -> monitoring screen:

At the bottom of this screen you will find the following jARM information:

Term	Meaning
jARM Overview	Summary of all jARM Requests.
Requests total	Number of requests executed so far.
Request rate	Requests per second since the jARM Monitoring has been started.
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.
jARM Requests	You can get the TOP 100 requests, i.e. the requests that needed the longest execution times. To restrict those entries you can specify two 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 'IDoc*'. You cannot make entries with the format '*XML*'.
max. listed Requests	Limits the number of requests to the given number. After pushing the button, the jARM Requests Overview is displayed.
jARM Components	You can get accumulated information about all components, that have been used in Requests. To restrict those entries you can specify an 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 '*Render*'. After pushing the button, the jARM Components Overview is displayed.

Mapping Inbound RFCs to Services

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

An Integration Server inbound call is an outbound call from the SAP system's point of view.

Creating a Function Module in an SAP System

Calling an Integration Server 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.

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 SAP GUI, go to the **ABAP Function Library. Choose Tools → ABAP Workbench** (transaction SE37).

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

3. Enter **`Z_B2B_PRODUCT`** in field **Function module**. This is the name of your SAP product retrieval function.

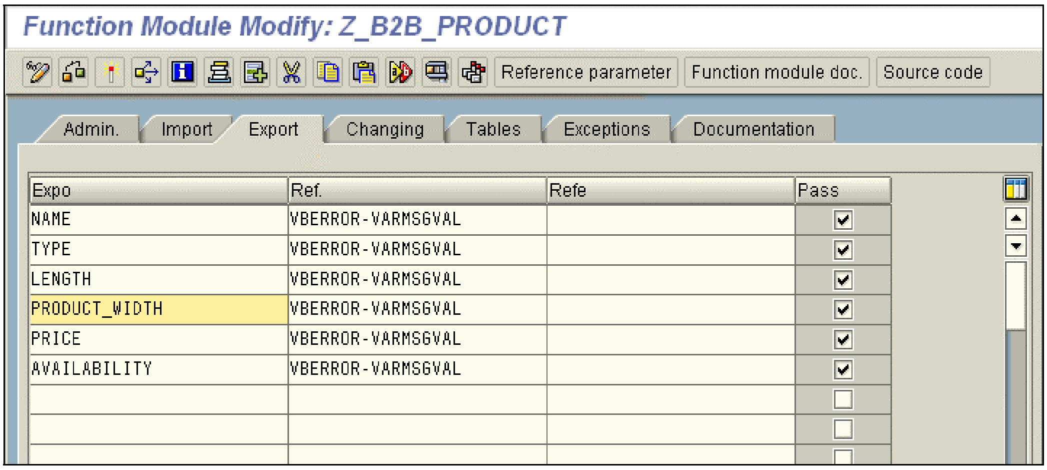
4. Choose **Create**.

5. Complete the following dialogs in accordance with the policies governing your SAP development environment. The only aspect relevant to the Integration Server is the field **Processing type**. Select **Remote Function Call supported** to allow this function to call externally to the Integration Server.

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

7. Add six exports: name, type, length,product_width, price, availability. Again, you must provide Reference fields for each of the exports. For this tutorial, use **`VBERROR-VARMSGVAL`** for these parameters.

8. Mark **Pass** for all parameters. The final screen should look like the following:



Expo	Ref.	Refe	Pass
NAME	VBERROR-VARMSGVAL		☑
TYPE	VBERROR-VARMSGVAL		☑
LENGTH	VBERROR-VARMSGVAL		☑
PRODUCT_WIDTH	VBERROR-VARMSGVAL		☑
PRICE	VBERROR-VARMSGVAL		☑
AVAILABILITY	VBERROR-VARMSGVAL		☑
			□
			□
			□

9. Save your function module and activate it.

Creating an Inbound Map

You need to associate a service on the Integration Server with the inbound RFC. The following steps map the service *tutorial.catalogue:getProductData* to inbound requests for Z_B2B_PRODUCT. (This service is provided with Integration Server as a sample flow service.)

To create an inbound map

1. Choose **SAP** from the Integration Server navigation panel.

2. Choose the **Lookup** tab.

3. Select the name of the SAP server on which the Z_B2B_PRODUCT function resides.

4. Enter `Z_B2B_PRODUCT` in the first **Function Name** field.

5. Choose **Lookup**. You will see a screen titled RFC Signature for Z_B2B_PRODUCT.

If not, you might not have defined your RFC Destination correctly. Review the steps in *Creating a RFC Destination on a SAP System* on page 5-2.

6. Follow the link **Define New Map** corresponding to an inbound request (SAP calling Integration Server).

7. On the resulting screen *RFC Map*, enter the following values. (Leave all other fields at their default values.)

Integration Server Folder/Service names are case sensitive, as are the parameters to the flow service.

In the field…	Specify…	Description
Generate for Listener	any listener specified and active	A drop down list with all listeners available for this server alias. This allows the generation of map signatures using the repository of the Listener which is selected for all metadata lookups

In the field…	Specify…	Description
		lookups.
Package	Default	This is the name of the package in which the service to which you are mapping the RFC is contained.
ACL	<None>	Access Control List.
Server Alias	(local)	This is the name of the target Integration Server
Folder	Demo	This is the name of the folder in which the service to which you are mapping the RFC is contained.
Service	Z_B2B_Product	This is the name of the service to which you are mapping the RFC.

8. Press **Save** to add this map.

Testing the Product Retrieval Function

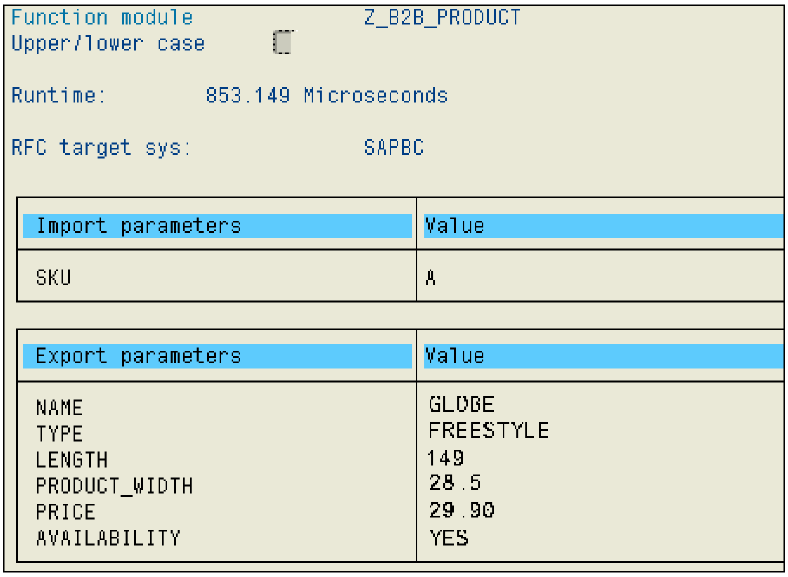
Use the following procedure to test the RFC you just created.

To test an inbound RFC

Use the following procedure to test the product retrieval function:

1. Ensure that an RFC Listener is running by doing the following:
    1. Choose **SAP** from the Integration Server navigation panel.
    2. Select the **SAP Servers** tab, if it is not already displayed.
    3. Select the Listeners for your SAP server.
    4. Verify that there are two green balls indicating that the listener is started and active. If not, either wait for the automatic listener restart or stop and start the listener by clicking on the balls in the **Started**? column. If no listeners listeners appear in the list, see *Configuring an RFC Listener in the Integration* Server on page 5-3.
2. In an SAP GUI session, go to *the* ABAP Workbench (transaction SE37) to test your new function module.
3. Enter **Z_B2B_PRODUCT** in field **Function module** and choose **Sngl. test**.
4. In the **RFC target system** field, type the Name of your RFC Destination (in this example, **SBCCER**).
5. In the **sku** field, enter a SKU. For this example, the SKUs that are available are A, B, C, D, or E. Case is not important.
6. Execute (**F8**) the RFC by selecting the appropriate toolbar button or selecting **Execute** from the **Function modules** menu.
7. You will receive the product data in your exports list similar as shown in the following figure.

If you receive an error from the Integration Server, make sure your map is correct and that the tutorial.catalogue:getProductData flow service is installed and functional.



Function module Z_B2B_PRODUCT
Upper/lower case □

Runtime: 853.149 Microseconds

RFC target sys: SAPBC

Import parameters	Value
SKU	A

Export parameters	Value
NAME	GLOBE
TYPE	FREESTYLE
LENGTH	149
PRODUCT_WIDTH	28.5
PRICE	29.90
AVAILABILITY	YES

Sending an RFC from a SAP System to the Integration Server

Normally, the RFCs sent to the Integration Server from an SAP System are mapped to a specific service. However, in some cases, you might want to *route* an RFC through the Partner Manager, but you can only route a mapped RFC if you provide a valid SBCHEADER with your function call. To route RFCs through the Partner Manager, you must include a header table that contains information that the Partner Manager uses to route the RFC.

Before the Partner Manager can receive an RFC, you must define an RFC listener for the Integration Server. For instructions, see *Configuring an RFC Listener in the Integration* Server on page 5-3.
To send an RFC to the Partner Manager, you must configure the SAP server to register the Integration Server as an RFC destination. For instructions on how to create an RFC Destination, see *Creating a RFC Destination on a SAP System* on page 5-2.

If no Inbound Map is defined, the SAP adapter will create a default Routing Rule based on the RFC attributes of the connection:

Message Type: Name of the function
Sender: System ID and Client Number
Receiver: Program ID of the Listener

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.

⚠ This table must not be defined in the function interface.

The name of the table must be SBCHEADER and it must have the following structure.

Component	Component type	DTyp	Length	Decimal places	Short text
NAME	SBCNAME	CHAR	32	0	SBC routing table, Keyfield
VALUE	SBCVALUE	CHAR	255	0	SBC routing table, value of Keyfield

This structure is defined in the SAP System starting with release 4.6A under the name SBCCALLENV. If you do not have this structure in your system, please define a similar one as above.

The header table can contain an arbitrary number of key/value pairs. If you want to pass additional keys to the SAP adapter you can define your own name/value pairs and insert them into the SBCHEADER table. You would read out these records inside a java module in the Integration Server with the following statements (case-sensitive):

```
Values sbcHeader = in.getValues("sbcHeader");
String property = sbcHeader.getString("fileName");
System.out.println("Property was "+property);
out.put("OUTPUT", property);
```

If you want the Partner Manager to invoke a routing rule to route the RFC, the header table must contain the sender and receiver for the RFC. When determining the routing rule to invoke, the Partner Manager uses the sender and receiver you specify in the header table and matches message types against the RFC name in place of a message type.

To have the Partner Manager use a routing rule, include the following information in the header table:

Key	Value
sender	Name of the sender. This name should match the name of a sender in the routing rule you want the Partner Manager to use to route the RFC.
receiver	Name of the receiving partner. This name should match the name of the receiver in the routing rule you want the Partner Manager 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 rule on the Integration Server), you can include transport information in the header table. The transport indicates where the Partner Manager is to route the incoming RFC. When the Partner Manager receives an RFC that specifies the transport, it does not invoke a routing rule but directly passes the RFC to the specified transport. The transports that you can identify in a header table to dynamically route an RFC through the Partner Manager are:

- Route the RFC to B2B service
- Route the RFC to an SAP server

- 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 B2B service, use the B2B service transport.

Key	Value
transport	B2B service. 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.
servicePath	Name of the interface in which the service resides. The interface name is case sensitive; use the exact combination of upper and lower case letters.
Service	Name of the service to which to pass the RFC. The service name is case sensitive; use the exact combination of upper and lower case letters.
valueScope	Where you want the Partner Manager to store the connection to the remote server. To save the connection in your own session, specify SESSION. Use SESSION when the work being performed requires state be maintained. To save the connection in a shared area, specify GLOBAL. Use GLOBAL when the work being performed is stateless.

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

Key	Value
transport	RFC. This value identifies the transport. Specify the value exactly as specified above
Server	Name of the SAP server to which you want the RFC routed.

- To post the RFC-XML to a URL, use the XML transport.

Key	Value
transport	XML. This value identifies the transport. Specify the value exactly as specified above
url	URL to which you want to post the RFC.
XmlType	The XML format you want the Partner Manager to use for the RFC. Specify *SAP-XML* if you want the RFC in an XML format that is compliant with the SAP XML specification. Specify **Values-XML** if you want the RFC in webMethods native XML format.
HttpUser	User name to supply for a user name/password authentification challenge (optionally)

Key	Value
httpPassword	Password to supply for a user name/password authentification challenge (optionally)

For information on sending RFCs as bXML, see page 7-10.

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. The SBCHEADER table cannot be added in the function builder and must **not** be part of the function interface of the module you want to call remotely.

The following example invokes the SBC_DEMO_COPY function module and echoes the inputs it receives in the INPUT parameter. The input DESTINATION parameter is interpreted as the RFC destination.

A wrapper for SBC_DEMO_COPY could look like this:

```
FUNCTION sbc_demo2.
*"----------------------------------------------------------------------
*"*"Lokale Schnittstelle:
*"     IMPORTING
*"          VALUE(INPUT) TYPE   CHAR255 OPTIONAL
*"          VALUE(DESTINATION) TYPE  CHAR32
*"     EXPORTING
*"          VALUE(OUTPUT) TYPE  CHAR255
*"     TABLES
*"           SBCHEADER STRUCTURE  SBCCALLENV
*"----------------------------------------------------------------------

CALL FUNCTION 'SBC_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 SBC_DEMO_COPY function module to the Integration Server and have the Partner Manager invoke a routing rule to route the message. When testing the function module from the SAP GUI, provide the following input values:

INPUT	Hello ...!
DESTINATION	SBCCER
SBCHEADER	Sender CERMAND receiver DELL

The SAP system routes the RFC to the specified RFC-Destination Integration Server (which corresponds to the name of the Integration Server listener). In the Integration Server, the Partner Manager invokes the routing rule that corresponds to the sender CERMAND, receiver DELL, and message type SBC_DEMO_COPY. To determine how the Partner Manager routes SBC_DEMO_COPY function module, you would need to inspect the corresponding routing rule that the Partner Manager invokes.

The following example illustrates how to route the SBC_DEMO_COPY function module to the Integration Server and dynamically specify the routing information. When testing the function module from the SAP GUI, provide the following input values:

INPUT	Hello ...!
DESTINATION	SBCCER
SBCHEADER	Transport RFC serverName CER

In this case, the SAP system routes the RFC to the specified RFC-Destination Integration Server. In the Integration Server, the gateway manger interprets the header table to determine how to route the RFC. The Partner Manager routes the message to the SAP server named CER using RFC. Note that CER must correspond to the name of an SAP server that is configured in the Integration Server.

Generating a Record for a Structure

This is a new feature introduced with SAP Adapter 4.6. It allows you to generate a record for the currently viewed structure. If a service or record with the same name is already existing in the server namespace, an exception is raised. Thus, you won't overwrite existing nodes.

Generation parameters:

- Folder: folder in which the record will be saved
- Record: name of the generated record
- Package: package in which the record will be saved

Click **Save** to generate a record with the given settings.

Working with 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, Email, 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 SAP adapter. Always use the option loadAs=bytes.

Convert the binary data using the Services pub.string:bytesToString or a combination of pub.web:stringToDocument (also accepts bytes) and documentToRecord.

Set the input parameter encoding to the mime Encoding in which the bytes have been encoded, i.e. one that is supported by the functions sun.io.ByteToCharXXX.class, where XXX stands for the encoding (for example, ASCII, ISO2022 or SJIS).

To Send Out Data via HTTP, FTP, Email, or Save It to File

- Ensure that you do not pass any String objects into the Services pub.client:http, pub.client:ftp, pub.client:smtp or save a String into a file.
- Call pub.string:stringToBytes with the correct encoding parameter.
- Pass the *bytes* into ftp/http or into your Service which writes the file. (This service should use java.io.FileOutputStream to write the file, not FileWriter.) That way the message is sent out of SAP adapter with proper encoding.

You can also use the ftp or smtp transport easily as standard transport from the Partner Manager. For further information, see *Example of Updating a Flow for FTP or Email Transport* on page 6-18.

To Encode Data from SAP Systems

Use the following procedure if you are using pub.sap.rfc:encode.

To create the standard SAP XML you would use the service `pub.sap.rfc:`*`encode`*, which handles encoding automatically.

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 recordToDocument:

1. In the recordToDocument Service add the attribute @encoding as a String as a child of the boundNode Record.
2. Insert the proper default value (for example, Shift-JIS, if your source data is Shift-JIS encoded).
3. In the recordToDocument Service add a record list for your parameters as a child of the boundNode Record.
4. Map your SAP parameters to the newly added record list. If you have several SAP parameters, you need to add a record set for each one.

Chapter 6: The Partner Manager

Introduction

This chapter describes the Partner Manager. It includes information on:

- Creating and maintaining routing rules
- Updating flow services that process routing rules
- Configuring the location of the Message Store
- Managing transactions in the Message Store

For specific information about how to send IDocs and RFCs to the Partner Manager to be routed, see *Routing IDocs and XML Messages to the Partner Manager* on page 8-1.

Overview of the Partner Manager

The Partner Manager is a built-in facility on the Integration Server that manages the routing of messages. The Partner Manager determines how and where to route a message based on *routing rules* that you define. Each routing rule is associated with an Integration Server flow service that the Partner Manager generates based on the information in the routing rule.

When the Partner Manager receives a message, it performs a routing rule lookup. After locating the routing rule for the incoming message, the Partner Manager invokes the flow service that is associated with the routing rule.

The Partner Manager records each message it receives as a transaction in the *Message Store*. By default, the Partner Manager maintains the Message Store in the xtn.log file in the *Integration Server-_directory*\packages\WmPartner\config directory. However, you can configure the Partner Manager to use a database for the Message Store.

You can view and manage the transactions that the Partner Manager records in its Message Store. You can re-send messages back into the Partner Manager for processing and delete transactions.

Routing Rules

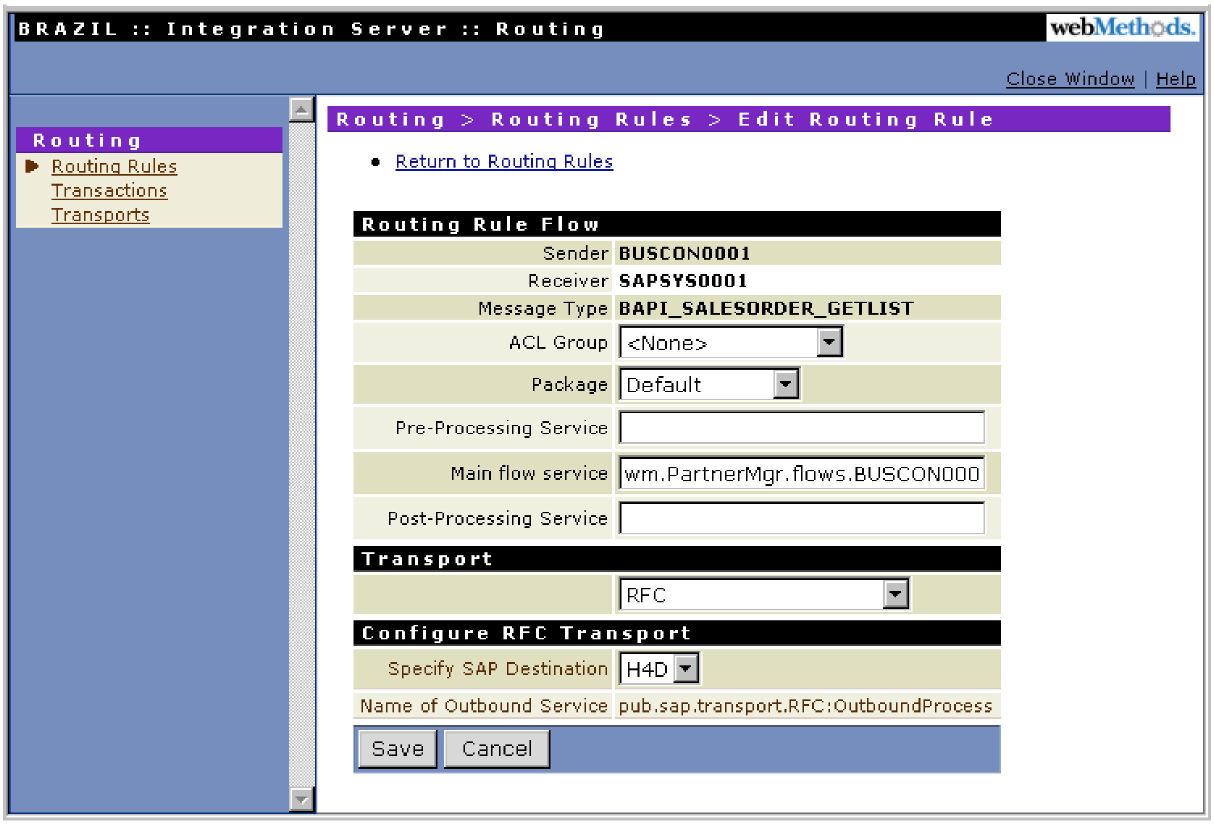
Routing rules indicate how a message is to be processed. Each routing rule is uniquely identified by its sender, receiver, and message type. When the Partner Manager receives a message, it performs a routing rule lookup to match the sender, receiver, and message type of the incoming message with the sender, receiver, and message type of routing rules.

If a matching routing rule is found, the Partner Manager processes the message as the routing rule indicates. If a routing rule is not defined for the message, the Partner Manager generates an incomplete routing rule. When you complete the rule, subsequent messages with the same sender, receiver, and message type are routed using the completed rule.

The routing rule 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 an Integration Server service, send a message via e-mail, FTP a message, route an IDoc to an SAP server via tRFC, route an RFC to an SAP server via RFC, or post an SAP IDoc- or RFC-XML to a URL.

Components of a Routing Rule

The following shows a routing rule.



Routing Rule Flow

Sender	BUSCON0001
Receiver	SAPSYS0001
Message Type	BAPI_SALESORDER_GETLIST
ACL Group	<None>
Package	Default
Pre-Processing Service	
Main flow service	wm.PartnerMgr.flows.BUSCON000
Post-Processing Service	

Transport

RFC

Configure RFC Transport

Specify SAP Destination	H4D
Name of Outbound Service	pub.sap.transport.RFC:OutboundProcess

A routing rule contains:

1 Sender, receiver, and message type

2 Name of the service that processes the routing rule

3 Pre-processing and post-procecessing service (optional)

4 How to route the message; that is, the transport

5 Additional Parameters based on the selected transport

Sender, Receiver, and Message Type

This information uniquely identifies a routing rule. The Partner Manager matches incoming messages to these fields to locate the routing rule to invoke to process the incoming message.

When a message is submitted, it must provide the Partner Manager 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 (e.g., a purchase order, a credit memo, an invoice, and so forth)

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 (see page 8-24).

For the message types ORDERS and ORDRSP a demo mapping is shipped with the SAP adapter.

sap.demo.idoc.mappings:orders

sap.demo.idoc.mappings:ordrsp

See *Constructing an IDoc with the IDoc Java API* on page 9-4 and Appendix D for the APIs to parse the IDoc.

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 rule. 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 rule for a specific company. For example, suppose that you want to execute a particular routing rule for all documents from XYZ Company. You don't 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.

Routing Rule Precedence

A routing rule 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 rule is matched and executed based on its value precedence. See the following table for the default match order. The most specific rule is matched (and executed) first.

A rule with these values is matched…	Sender	Receiver	msgType
First	arbitrary string	arbitrary string	arbitrary string
Second	*	arbitrary string	arbitrary string
Third	arbitrary string	*	arbitrary string
Fourth	arbitrary string	arbitrary string	*
Fifth	*	*	arbitrary string
Sixth	*	arbitrary string	*
Seventh	arbitrary string	*	*
Eighth (last)	*	*	*

If rule matching is performed by an external service via a getRule service, then the value precedence is dependent on your custom implementation. A custom getRule service uses wm.PartnerMgr.gateway.getRule:getRuleSpecification to bypass the default routing rule functionality and returns the routing rule that you want invoked. The getRule service must be specified in the server.cnf file on the watt.WmPartners.getRule parameter. Any kind of wildcards can be implemented within the getRule service.

Example 1

Suppose that you have a routing rule with these values:

- Sender = snd
- Receiver = receiver
- msgType = msg1

A message arrives with these values:

- Sender = snd1
- Receiver = receiver
- msgType = msg1

No routing rule exists with those exact values, so the message is matched to a routing rule with the following pattern:

- Sender = *
- Receiver = arbitrary string

- msgType = arbitrary string

Example 2

Suppose you have a routing rule with these values:

- Sender = snd
- Receiver = receiver
- msgType = msg1

A message arrives with these values:

- Sender = snd1
- Receiver = receiver
- msgType = msg2

No routing rule exists with those exact values, so the message is matched to a routing rule with the following pattern:

- Sender = *
- Receiver = arbitrary string
- msgType = *

To override the default match order, see *Overriding Routing Rule Match Order* on page.6-7.

When an Inbound Message Contains a Wildcard

If an inbound message contains an asterisk (*) for its receiver or msgType parameters, the Partner Manager creates a mailbox directory on the server, replacing "*" with "WildCard". This is necessary because file systems do not allow special characters for directory or file names.

For example, suppose you have a routing rule with these values:

- Sender = *
- Receiver = Rcv
- msgType = *

If a message arrives with these values:

- Sender = sender
- Receiver = Rcv

- msgType = *

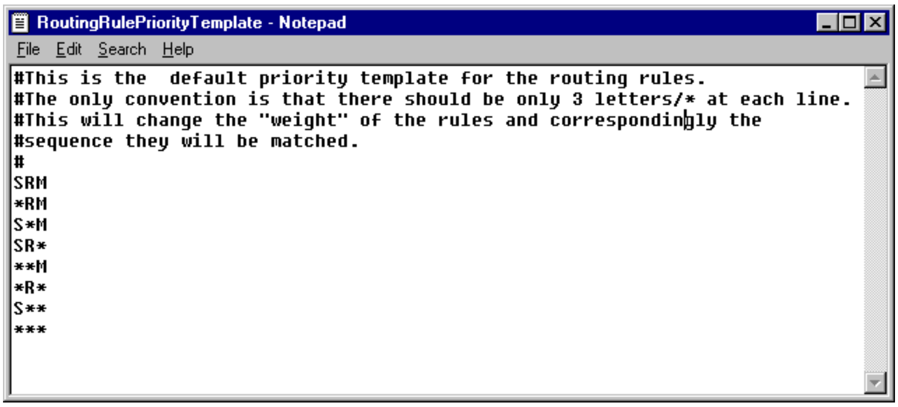
The corresponding .values file is stored in the mailbox at *Integration Server_directory*\packages\WmPartners\pub\mailbox\Rcv\WildCard.

Overriding Routing Rule Match Order

When routing rules contain wildcards for the Sender, Receiver, or Message Type, there is a default order in which the routing rule is matched to the incoming document's values. (For details, see "Routing Rule Precedence" on page 6-4.) You can override this default order by modifying the RoutingRulePriorityTemplate.cnf file in your WmPartners package. See the following procedure.

To override routing rule match order

1. In a text editor such as Notepad, open the RoutingRulePriorityTemplate.cnf file, located in the *Integration Server_directory*\packages\WmPartners\config directory.



```
#This is the  default priority template for the routing rules.
#The only convention is that there should be only 3 letters/* at each line.
#This will change the "weight" of the rules and correspondingly the
#sequence they will be matched.
#
SRM
*RM
S*M
SR*
**M
*R*
S**
***
```

2. Edit the file to reflect the match order priority that you want for rules containing wildcards. See the following table.

This character in the file…	Represents
#	A comment. Put at the beginning of a line that contains a comment.
S	The *sender* value.
R	The *receiver* value.
M	The *msgType* value.
*	A wildcard. It will match any value.

3. Save the file.

4. In the Server Administrator, under Packages, click Management.

5. Next to WmPartners, click Reload . The new routing rule match order is now in effect.

If rule matching is performed by an external service via a getRule service, then the value precedence is dependent on your custom implementation. A custom getRule service uses wm.PartnerMgr.gateway.getRule:getRuleSpecification to bypass the default routing rule functionality and returns the routing rule that you want invoked. The getRule service must be specified in the server.cnf file on the watt.WmPartners.getRule parameter. Any kind of wildcards can be implemented within the getRule service.

Name of the Service that Processes the Routing Rule

You specify the name you want the Partner Manager to use for the flow service that processes the routing rule. The Partner Manager generates the flow service based on the fields you specify in the routing rule. You select the package in which this flow service resides when you create the routing rule.

When an incoming message has a sender, receiver, and message type that match the sender, receiver, and message type of this routing rule, the Partner Manager triggers the routing rule. When the routing rule is triggered, the Partner Manager invokes this flow service, which controls the processing for routing the message.

If the routing rule identifies a pre-processing service, this flow service passes the incoming message to the pre-processing service. After the pre-processing service is completed, this flow service gains control again and invokes the built-in outbound service for the transport.

If the routing rule does not identify a pre-processing service, this service simply invokes the built-in outbound service that is associated with the transport.

After you create a routing rule and the Partner Manager generates the flow service, you can update the flow service using the webMethod Developer. For more information, see *Updating the Flow Associated with a Routing Rule* on page 6-18.

Pre-Processing Service

Optionally, you can specify a pre-processing flow, Java, or C/C++ service. The flow service that processes the routing rule invokes the pre-processing service before it invokes the built-in service that handles the selected transport.

The pre-processing service is passed the incoming message. It can perform any processing, including modifying the message to be routed. When the pre-processing service is complete, the flow service that processes the routing rule invokes the built-in service that handles the selected transport.

Post-Processing Service

When you define a routing rule, you can optionally instruct the Partner Manager to execute a post-processing service after routing a document to its destination. When you specify the name of a post-processing service in the routing rule, the Partner Manager executes that service after the Outbound Process is complete in the routing service.

A post-processing service is a service that you create to perform any work that must be executed immediately after the Partner Manager sends a document out.

The use of these services is purely optional.For example, you might use a post-processing service to put a copy of the PO in a database?.

How to Route the Message (Transport)

The transport indicates how the Partner Manager is to route the message (for example, route it to a service or FTP it to a host). The transport can be one of the following:

B2B service	Routes the message to an service on the local Integration Server or on a remote Integration Server.
Email Outbound Service	Sends the message in an e-mail message.
FTP Outbound Service	FTPs a message to a host.
ALE (R/3 IDoc)	Routes an IDoc to an SAP server via tRFC.
RFC	Routes an RFC to an SAP server via RFC or tRFC.
XML	Posts an SAP bXML, IDoc- or RFC-XML to a URL.
BAPI	Routes a BAPI to an SAP Server via RFC or tRFC

Based on the transport you select, the Partner Manager displays additional fields required by the selected transport.

Establishing Routing Rules

You use the *Routing Rules* screens to establish routing rules for each message that the Partner Manager is to process. Use the following procedure to define a routing rule.

Routing has been enhanced: A new transport *BAPI* has been added, which is able to dispatch synchronous and asynchronous BAPI calls sent from the Web via XML

The existing *XML* transport has been extended to support the new XML features introduced with this extension package. A new XML dialect *bXML* is now available, which is BizTalk based and can be applied to BAPI and RFC calls. Details on how to configure the XML transport can be found in the chapter *Setting up Routing for BAPIs*

To define a routing rule

1. Choose the Routing task from the Integration Servernavigation panel.
2. Select the **Routing Rules** link, if is not already selected.

   The Partner Manager displays the View Routing Rules screen. The top part of the screen lists existing rules.
3. Below the list of existing rules are three input boxes. Type the name of the sender in the first box, the name of the receiver in the second box, and the message type in the third box.
4. Select **Add Rule**. The Partner Manager displays the Routing Rule screen.
5. Set the Flow Information parameters as follows:

Key	Value
Name	Name you want to use for the flow service that is invoked by this routing rule. By default, the Partner Manager creates a name that incorporates the sender, receiver, and message type information. If you are satisfied with the default name, leave the name as is.
Package Location	Package for the flow service. Select the package in which you want the service to reside from the drop down list.

The Integration Server creates a file using the name you specify. If the sender, receiver, or message type values contain characters that your file system does *not* support, be sure to update the name that the Partner Manager creates. For example, if the message type is XML\PurchaseOrder, you would want to change the ‘ \ ‘ to a different character (for example, XML_PurchaseOrder).

6. If you want to use a pre-processing service, specify the fully qualified name of the service in the **Invoke Pre-Processing Service/Flow** field. The service name is case sensitive. Be sure to use the exact combination of upper and lower case letters when specifying the service name.

7. 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, see the page indicated below.

Select this transport…	To
B2B service	Route the message to an Integration Server service.
Email Outbound Service	Send the message in an e-mail message.
FTP Outbound Service	FTP the message to a host.
ALE (R/3 IDoc)	Route an IDoc to an SAP server
RFC	Route an RFC to an SAP server
XML	Posts an SAP bXML, IDoc or RFC-XML and any arbitrary XML document to a URL.
BAPI	Routes a BAPI to an SAP server

8. Press **Save**.

9. Enable the routing rule by clicking on the enabled column

Transport: B2B service

Use this transport to route messages to another service that executes on the local machine or on a remote Integration Server.

The entire pipeline to the specified B2B service. When a document is routed with this transport, it doesn’t make any difference where you put it in the pipeline, because the entire pipeline is transmitted to the destination service.

Set the Configure B2B service parameters as follows:

Key	Value
Server Alias	Integration Server on which the service to invoke resides. If the service resides on the same service as the Partner Manager that is routing the message, select **(local)**. If the service resides on a remote server, select the name of the Integration Server from the drop down list. The drop down list contains the names of remote Integration Servers that you have defined. For information about defining aliases for remote Integration Servers, see the *webMethods Integration Server Administrator's Guide*.
Folder	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 as defined on the Integration Server that you specified in the **Server Alias** field.
Service	Name of the service to which the message is to be passed. The service name is case sensitive; use the exact combination of upper and lower case letters as defined on the Integration Server that you specified in the **Server Alias** field.
Scope	Whether you want the connection information for the server to be stored in your session or in a shared location. If the work being performed requires that state be maintained, select **SESSION**. If the work being performed is stateless, select **GLOBAL**.

When sending an IDoc or an RFC between two Integration Servers using B2B service Transport, enter `wm.PartnerMgr.gateway.transport.B2B` as Folder and `InboundProcess` as Service

Transport: Email Outbound Service

Use this transport to e-mail a message to a trading partner.

If you use this transport, you must update the flow service that the Partner Manager generates for the routing rule. For information about how to update the routing rule, see *Example of Updating a Flow* on page 6-18.

Transport the contents of data/content to the specified email address, where *data* is a Record (a Values object) and content is an object element of data. If your document is to be routed through the email transport:

- It must be in the form of a String, a byte [ ], or an InputStream.
- You must map the document to *data*/*content* before invoking the Partner Manager.

Set the parameters as follows

Key	Value
SMTP Host	Host name of the SMTP server to which you want to send the e-mail message.
Content Type	Content type the Partner Manager should specify for the e-mail message. For example, specify application/x-edi-message for an EDI message.

From	E-mail address that you want the Partner Manager to use for the sender of the e-mail message, for example, orderingsystem@buyer.com.
To	E-mail address to which you want the Partner Manager to send the e-mail message, for example, orders@seller.com.
CC	Optionally, the e-mail address to which you want the Partner Manager to send a complimentary copy of the e-mail message, for example, accountspayable@buyer.com.
BCC	Optionally, the e-mail address to which you want the Partner Manager to send a blind complimentary copy of the e-mail message, for example, manager@buyer.com.
Subject	Subject you want the Partner Manager to use for the e-mail message.

Transport: FTP Outbound Service

Use this transport to FTP a message to a location.

⚠ If you use this transport, you must update the flow service that the Partner Manager generates for the routing rule. For information about how to update the routing rule, see *Example of Updating a Flow* on page 6-18.

Transport the contents of *data*/*content* to the specified file location, where *data* is a Record (a Values object) and content is an object element of *data*. If your document is to be routed through the FTP transport:

- It must be in the form of a String, a byte [ ], or an InputStream.
- You must map the document to *data*/*content* before invoking the Partner Manager.

Set the Configure FTP Outbound Service parameters as follows:

Key	Value
Host Name	Host name of the remote FTP server.
Port Number	Port number on which the remote FTP server listens for incoming requests.
User Name	User name the Integration Server must supply to connect to the remote FTP server.
Password	Password the Integration Server must supply to connect to the remote FTP server.
File Path	Directory path and file name on the target machine where the message is to be placed.

Transport: ALE(R/3 IDoc)

Use this transport to route an IDoc to an SAP server via tRFC.

Before you can route an IDoc to an SAP server, you must define the SAP server. For instructions, see *Defining SAP Servers* on page 4-3.

Set the Configure ALE Transport parameters as follows:

Key	Value
Specify SAP Destination	Name of the SAP server to which you want the Partner Manager to route the IDoc. Select an SAP server from the drop down list. The drop down list contains the names of servers that are defined to the Integration Server. For instructions on how to define an SAP server to the Integration Server, see *Defining SAP Servers* on page 4-3.

Transport: RFC

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

Set the Configure RFC Transport parameters as follows:

Key	Value
Specify SAP Destination	Name of the SAP server to which you want the Partner Manager to route the RFC. Select an SAP server from the drop down list. The drop down list contains the names of servers that are defined to the Integration Server. For instructions on how to define an SAP server to the Integration Server, see *Defining SAP Servers* on page 4-3.

Transport: XML

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

Set the Configure XML Transport parameters as follows

Key	Value
Specify URL	URL to which you want the Partner Manager to post the XML message.
XML dialect	Choose between the XML dialects SAP-XML, bXML, Values-XML or SOAP XRFC (XRFC with SOAP envelope). If you select SAP-XML the content type is set to ‘application/x-sap.idoc’ (respectively ‘.rfc’). Therefore the receiving server has to understand this content type. If ‘Arbitrary XML’ is selected, the transport expects the XML document as string in the variable xmldata.
Use text/xml as content type	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.
Use utf-8 as encoding	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.
Use Bapi Format	Set this value to *ON* if you want to use the BAPI XML format. (This field is only active when using the bXML dialect.)
Business Object	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.)
BAPI	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 (optional)	An optional parameter that allows you to specify a username fo authentification on the remote Web system.
httpPassword (optional)	An optional parameter that allows you to specify a passwordfo authentification on the remote Web system.

When sending an IDoc between two Integration Servers via XML transport, use http://*Integration Server host*:*port*/invoke/pub.sap.transport.ALE/InboundProcess as URL.

When sending an RFC to a second Integration Server without an outbound map over the Partner Manager use http://*Integration Server host*:*port* /invoke/pub.sap.transport.RFC/InboundProcess as URL.

Transport: BAPI

Set the Configure BAPI Transport parameters as follows

Key	Value	
Specify SAP Destination	Name of the SAP server to which you want the Partner Manager to route the RFC. Select an SAP server from the drop down list. The drop down list contains the names of servers that are defined to the Integration Server. For instructions on how to define an SAP server to the Integration Server, see *Defining SAP Servers* on page 4-3.	
Processing restrictions	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 and 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 an an XML error message is returned.

When a Routing Rule Is Not Defined

When the Partner Manager receives a message for which it cannot locate a routing rule, it logs the message in its Message Store and throws an exception.

In addition, the Partner Manager generates an incomplete routing rule for the message. An SAP administrator should complete the routing rule definition. Subsequent messages with the same sender, receiver, and message type are automatically routed using the completed rule.

The following procedure describes how to complete the routing rule.

To complete a routing rule that is “Not Ready” (indicated by a red ball)

1. Select the Routing task from the Integration Server Administrator navigation panel.

2. Select the **Routing Rules** tab, if it is not already selected. The Partner Manager displays the View Routing Rules screen. The top part of the screen lists existing rules. The incomplete routing rule has the status “Not Ready” (indicated by the red ball).

3. Click ✔ for the rule you want to complete. The Partner Manager displays the Routing Rules screen.

4. Set the Flow Information, Invoke Pre-Processing, and Transport parameters as described in *Establishing Routing Rules* on page 6-9.

5. Choose **Save**. The next time the Partner Manager receives a message that has the sender, receiver, and message type identified in the routing rule that you just completed, the Partner Manager will trigger this routing rule.

6. Enable the routing rule by pressing. The next time the Partner Manager receives a message that has the sender, receiver, and message type identified in the routing rule that you just completed, the Partner Manager will trigger this routing rule.

Managing Routing Rules

After you define routing rules, you can:

- Display the routing rules that have been established
- Update information for routing rules
- Delete routing rules

Displaying Routing Rules

Perform the following procedure to view the existing routing rules.

To view routing rules

1. Select the Routing task from the Integration Server navigation panel.
2. Choose the **Routing Rules** tab on the bottom to open the View Routing Rules screen.

Updating Routing Rules

After you create a routing rule, you can update information for the routing rule if necessary. You cannot change the sender, receiver, or message type information for a routing rule.

When you update the routing rule, the Partner Manager updates the flow service that processes the routing rule. The changes you make to the routing rule through the Routing Rule screen affect only the first operation in the flow service. If you have added additional operations to the flow, they will not be altered. For more information about updating the flow service, see *Updating the Flow Associated with a Routing Rule* on page 6-18.

To edit a routing rule

1. Select the Routing task from the Integration Server navigation panel.
2. Choose the **Routing Rules** tab on the bottom to open the View Routing Rules screen.
3. Click ✓ in the **Edit** column for the routing rule you want to edit. The Partner Manager displays the Routing Rule screen.
4. Make changes as necessary. For information on what you can specify, see *Establishing Routing Rule* on page 6-9.
5. After you make your changes, choose **Save**.

Disabling Routing Rules

Use the following procedure to disable a routing rule. When you disable a routing rule, all routing service and rule information is preserved until you re-enable it.

To disable a routing rule

1. Start Server Administrator. See the *webMethods Integration Server Administrator's Guide* if you need procedures for this step.

2. On the **Adapters** menu, click **Routing**. The View Routing Rules screen appears.

3. Locate the rule that you want to disable. Under **Enabled?**, click ✓ **Yes** until it turns to **No**.

Deleting Routing Rules

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

When you delete a routing rule, the Partner Manager deletes the flow service that is associated with the routing rule.

To delete a routing rule

1. Click **Routing** from the Integration Server navigation panel.

2. Click the **Routing Rule** tab.

3. Locate the routing rule you want to delete, and click ✕ in the **Delete** column.

Editing a Routing Service

When necessary, you can edit a routing service with webMethods Developer to incorporate additional operations before or after the transport delivers the document. For example, you might want to insert flow-control operations (for example, LOOP or BRANCH operations) around the pre-processing service, or you might want to include some error-handling operations immediately after the transport service.

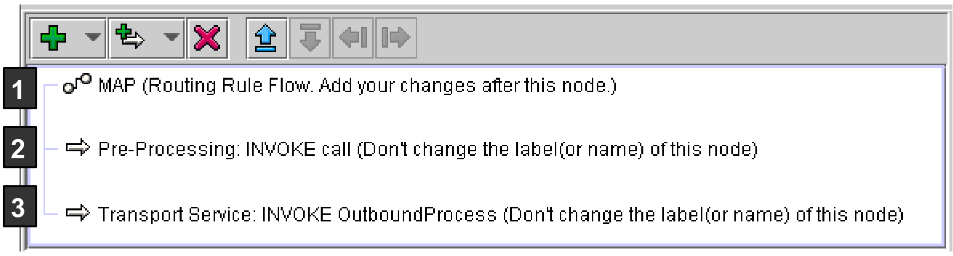
Guidelines to Follow When Editing a Routing Service

When you edit a routing service, keep the following points in mind:

- Do not add any flow operations before the first MAP operation.

- Do not modify the pre-processing service's label property (if a pre-processing service exists in the flow).

- Do not modify the transport service's label property.



1 MAP (Routing Rule Flow. Add your changes after this node.)

2 Pre-Processing: INVOKE call (Don't change the label(or name) of this node)

3 Transport Service: INVOKE OutboundProcess (Don't change the label(or name) of this node)

Updating the Flow Associated with a Routing Rule

When you create a routing rule, the Partner Manager generates a flow service. If you want, you can modify the flow service that the Partner Manager generates. For example, you might want to update the flow service to digitally sign a message before it is routed. If you use the FTP Outbound Service or Email Outbound service transport, you must update the flow service to map the input parameters.

Use Developer to update the flow service. Be sure to abide by the following rules for updating the flow service:

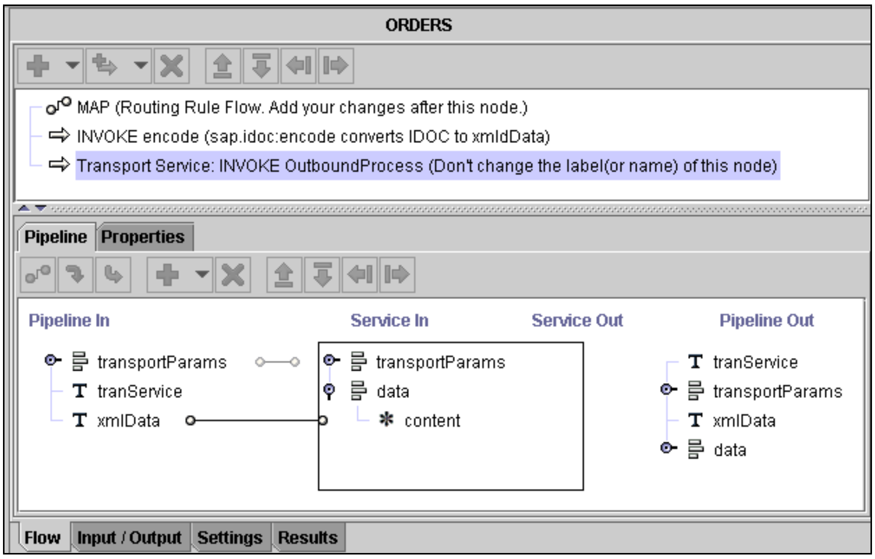
- Do not add any flow operations before the first MAP flow operation.
- Do not update the flow operation for the pre-processing service, if it exists.
- Do not update the flow operator for the outbound transport service.

Example of Updating a Flow for FTP or Email Transport

When you create a routing rule that uses the FTP Outbound Service or Email Outbound Service transport, you must update the flow created for the routing rule to make the flow service functional. From Developer, make the following changes to the routing rule flow to make it functional.

To update the flow for FTP or Email Transport for sending an IDoc

1. Add a flow operation to invoke the pub.sap.idoc:encode service after the first MAP flow operation. This service converts the IDoc to an XML string, which is the format the FTP and Email Outbound Transport service can understand.
2. Select the last flow operation, Transport Service: INVOKE Outbound Process.
3. Select the **Pipeline** tab if it is not already selected.
4. Expand the data variable in **Service In** to display its structure.
5. Add a map to link the xmlData variable in **Pipeline In** to the content variable in **Service In**.



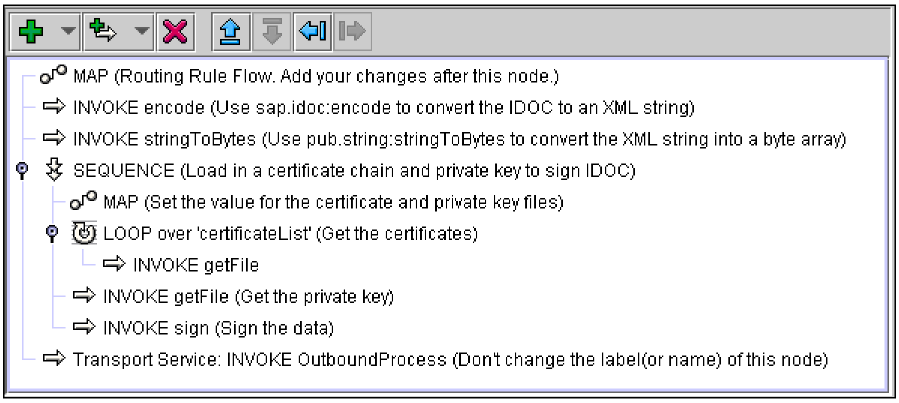
```
MAP (Routing Rule Flow. Add your changes after this node.)
INVOKE encode (sap.idoc:encode converts IDOC to xmlData)
Transport Service: INVOKE OutboundProcess (Don't change the label(or name) of this node)
```

```
Pipeline In          Service In          Service Out     Pipeline Out
transportParams      transportParams                     tranService
  tranService        data                                transportParams
  xmlData              content                           xmlData
                                                         data
```

6. Save the flow.

Example of Digitally Signing a Message

This shows a flow that has been updated to digitally sign a message before it is routed to its destination by the transport service.



```
MAP (Routing Rule Flow. Add your changes after this node.)
INVOKE encode (Use sap.idoc:encode to convert the IDOC to an XML string)
INVOKE stringToBytes (Use pub.string:stringToBytes to convert the XML string into a byte array)
SEQUENCE (Load in a certificate chain and private key to sign IDOC)
    MAP (Set the value for the certificate and private key files)
    LOOP over 'certificateList' (Get the certificates)
        INVOKE getFile
    INVOKE getFile (Get the private key)
    INVOKE sign (Sign the data)
Transport Service: INVOKE OutboundProcess (Don't change the label(or name) of this node)
```

Configuring the Location of the Message Store

The Integration Server uses the *Message Store* to keep track of all messages that pass through the Partner Manager.

By default, the Partner Manager maintains the Message Store in the xtn.log file in the *Integration Server_directory*\packages\WmPartner\config directory. If you want the Partner Manager to maintain the Message Store in the local file system, you do not need to perform any further configuration. However, if you want the Partner Manager to use a JDBC-compliant database for the Message Store, you need to configure the Partner Manager to identify the location of the database.

The Partner Manager supports JDCB-compliant RDBMS packages, such as SAP Adabas, Oracle 6.0 and higher, Informix 7.0 and higher, and MS SQL Server 6.50.201.

Do *not* use a database with either the Microsoft or SUN built-in JDBC-ODBC bridge as it can cause problems with the Integration Server.

This section describes how to set up the Partner Manager to use the following databases:

Database	Refer to page
Microsoft SQL Server 7.x	6-20
Oracle 8i	6-22
SAP ADABAS	6-24

Microsoft SQL Server 7.x

To configure the Partner Manager to use a SQL Server 7.x database for the Message Store, perform the following procedure.

To use an SQL 7.x database for the Message Store

1. Perform the following steps to set up the SQL database:

    1. Obtain a pure JDBC driver for the SQL database, for example, weblogic.jdbc.mssqlserver4.Driver from WebLogic or drivers from vendors, such as, Intersolve or Imprise.

The Microsoft SQL Server 7.x ODBC driver is *not* compatible with JDBC.

    2. Install the JDBC driver that you obtained for the database in the classpath. Follow your vendor's JDBC installation instructions.

    3. Either set up the database so the Partner Manager can add tables or add the required tables to the database.

        To let the Partner Manager update the database, create a user account that has privileges to create database tables. In Step 1, you will configure the Integration Server to use this user account to log into the database. When you configure the database alias for this database, you will identify the user account in the **DB Username** field. Also, create a user-defined type named *Date* that has the data type "datetime."

        If you do not want the Integration Server to have the privilege to create database tables, have your database administrator create the following tables for the database:

Table	Column	Type	Length
WMTRANSACTION_Log	tid	VARCHAR	50

Table	Column	Type	Length
g	creationdate	DATE	
	message	VARCHAR	255
WMTRANSACTIONS	tid	VARCHAR	50
	sender	VARCHAR	50
	receiver	VARCHAR	50
	msgType	VARCHAR	50
	package	VARCHAR	50
	docURI	VARCHAR	50
	flow	VARCHAR	50
	lastError	VARCHAR	50
	state	INTEGER	
	wmTime	DATE	

2. Perform the following steps to set up the Integration Server:

   1. Configure the Integration Server to have a database alias for the database. For instructions, see the *webMethods Integration Server Administrator's Guide*.

   2. When defining the alias, specify the following information in the Details section of the New DB Alias screen:

Key	Value
Alias	transactions. You must specify this name using all lower case.
DB URL	URL for the database, for example, jdbc:weblogic:mssqlserver4:edi:1433.
DB Username	User name the Integration Server must supply to log into the specified database. The Partner Manager requires certain tables to exist in the database. When the Integration Server connects to the database, the Partner Manager attempts to create the tables if they do not exist. In Step 3, you should have either defined a user account for your database that can add tables, or created the tables that the Partner Manager requires. If the Partner Manager is to create the tables, specify the user account that has the database privilege required to create database tables in the **DB Username** field.

DB Password	Password the Integration Server must supply to log into the database.
DB Driver	The name of the Java class for the JDBC driver, for example, weblogic.jdbc.mssqlserver4.Driver.

3. Enable database logging. To enable logging, shut down Integration Server and add the watt.PartnerMgr.xtn.store=db paramter to the server.cnf file, located in the *Integration Server_directory* \config directory.

Oracle 8i

To configure the Partner Manager to use an Oracle 8i database for the Message Store, perform the following procedure.

To use an Oracle 8i database for the Message Store

1. Perform the following steps to set up the Oracle database:

   1. Install the Oracle 8I thin driver’s classes12.zip in the classpath.

   2. Either set up the database so the Partner Manager can add tables or add the required tables to the database.

      To let the Partner Manager update the database, create a user account that has privileges to create database tables. In Step 2, you will configure the Integration Server to use this user account to log into the database. When you configure the database alias for this database, you will identify the user account in the **DB Username** field.

      If you do not want the Integration Server to have the privilege to create database tables, have your database administrator create the following tables for the database:

Table	Column	Type	Length
WMTRANSACTION_Log	tid	VARCHAR	50
	creationdate	DATE	
	message	VARCHAR	255
WMTRANSACTIONS	tid	VARCHAR	50
	sender	VARCHAR	50
	receiver	VARCHAR	50
	msgType	VARCHAR	50
	package	VARCHAR	50

Table	Column	Type	Length
	docURI	VARCHAR	50
	flow	VARCHAR	50
	lastError	VARCHAR	50
	state	INTEGER	
	WmTime	DATE	

2. Configure the Integration Server to have a database alias for the database. For instructions, see the *webMethods integration Server Administrator’s Guide*. When defining the alias, specify the following information in the Details section of the New DB Alias screen:

Key	Value
Alias	transactions. You must specify this name using all lower case.
DB URL	URL for the database, for example, jdbc:oracle:thin:*username*/ *password@hostname:port-number:DatabaseName* where: *username* is the user name to supply to log into the server. *password* is the password to supply to log into the server. *hostname* is the host name of the Oracle 8I server. *portnumber* identifies the port on which the Oracle 8I server listens for incoming requests. *DatabaseName* is the name of the database.
DB Username	User name the Integration Server must supply to log into the specified database. The Partner Manager requires certain tables to exist in the database. When the Integration Server connects to the database, it attempts to create the tables if they do not exist. In Step 2, you should have either defined a user account for your database that can add tables or created the tables that the Partner Manager requires. If the Partner Manager is to create the tables, specify the user account that has the database privilege required to create database tables in the **DB Username** field.
DB Password	Password the Integration Servermust supply to log into the database.

DB Driver	The name of the Java class for the JDBC driver, for example, oracle.jdbc.driver.OracleDriver.

3. Enable database logging. To enable logging, shut down Integration Server and add the watt.PartnerMgr.xtn.store=db parameter to the server.cnf file, located in the *Integration Server_directory* \config directory.

4. If you want to use an Oracle DB of release 8i, you have to disable statement caching. To disable statement caching, shut down Integration Server and add the watt.server.jdbc.statementCache=false parameter to the server.cnf file, located in the *Integration Server_directory* \config directory.

SAP ADABAS Database

To configure the Partner Manager to use the SAP ADABAS database for the Message Store, perform the following procedure.

The JDBC driver for the SAP ADABAS database (sapdbc.jar), is located in the *Integration Server_directory \packages\SAP\code\jars* directory. The Integration Serverwill automatically load the JDBC driver.

To use an SAP ADABAS database for the Message Store

1. Either set up the database so the Partner Manager can add tables or add the required tables to the database.

    1. To let the Partner Manager update the database, create a user account that has privileges to create database tables. In Step 2, you will configure the Integration Server to use this user account to log into the database. When you configure the database alias for this database, you will identify the user account in the **DB Username** field.

    2. If you do not want the Integration Server to have the privilege to create database tables, have your database administrator create the following tables for the database:

Table	Column	Type	Length
WMTRANSACTION_Log	tid	VARCHAR	50
	creationdate	TIMESTAMP	
	message	VARCHAR	255
WMTRANSACTIONS	tid	VARCHAR	50
	sender	VARCHAR	50
	receiver	VARCHAR	50
	msgType	VARCHAR	50
	package	VARCHAR	50
	docURI	VARCHAR	50

Table	Column	Type	Length
	flow	VARCHAR	50
	lastError	VARCHAR	50
	state	INTEGER	
	WmTime	TIMESTAMP	

2. Configure the Integration Server to have a database alias for the database. For instructions, see the *webMethods integration Server Administrator's Guide*.

   When defining the alias, specify the following information in the Details section of the New DB Alias screen:

Key	Value
Alias	transactions. You must specify this name using all lower case.
DB URL	URL for the database, for example, jdbc:sapdb://hostname/dbname, where hostname is the name of the machine on which the database is running and dbname is the name of the database, which is defined when the database is installed. If you want the database to write a trace file, use a URL with the format jdbc:sapdb://*host*/dbname?trace=path where path identifies where you want the trace file recorded, for example, C:\temp\sapdbc.txt.
DB Username	User name the Integration Server must supply to log into the specified database. The Partner Manager requires certain tables to exist in the database. When the Integration Server connects to the database, the Partner Manager attempts to create the tables if they do not exist. In Step 1, you should have either defined a user account for your database that can add tables or created the tables that the Partner Manager requires. If the Partner Manager is to create the tables, specify the user account that has the database privilege required to create database tables in the ***DB Username*** field, for example, SAPR3.
DB Password	Password Integration Server must supply to log into the database.
DB Driver	The name of the Java class for the JDBC driver, for example, com.sap.dbtech.jdbc.DriverSapDB.

3. Enable database logging. To enable logging, you must shut down Integration Server and add the watt.PartnerMgr.xtn.store=db parameter to the server.cnf file, located in the *Integration Server_directory* \config directory.

Managing Transactions in the Message Store

The Partner Manager logs each message it receives as a transaction in the Message Store. Each message is associated with a transaction ID that uniquely identifies the transaction. If the Partner Manager receives a message without a transaction ID, it creates one for the message.

Use the Transactions screens to view and manage the transactions that the Partner Manager records in its Message Store. Here you can view transactions and their processing log as well as delete transactions that are no longer needed.

The Administrator should manually follow-up transactions in status 'Rolled back' as follows:

Try to find and eliminate the reason of failure and then resend the transaction from SM58, if it originally came from an SAP system, then ask your partner to resend the transaction, if it originally came in via FTP or HTTP.

Viewing Transactions in the Message Store

You can view transactions that the Partner Manager currently has in its Message store. Perform the following procedure to view a list of transactions.

To view the transactions in the Message Store

1. Select the **Routing** task from the Integration Server navigation panel.

2. Select the **Transactions** tab if it is not already selected. The Partner Manager displays the Transactions screen that lists the transactions.

3. To view detail information for a transaction, press on a transaction id in the **TID** column.

   The Transaction list allows you to view the details of each transaction, including the sender, receiver, message type, transaction ID, current state, last error (if any), and an audit trail of services that have processed the message.

   The **Current State** field contains the transaction state. The information in this field is valid only when routing IDocs with tRFC. The following describes the valid states of a tRFC transaction:

Key	Value
Created	The transaction has just been created and is ready for execution.
Executed	The transaction has completed execution.
Rolled Back	Execution of the transaction has failed. The transaction is ready for execution again.
Committed	Execution of the transaction was successful. The transaction is waiting for confirmation.
Confirmed	The transaction is complete.

If you want, you can view the message in XML, HTML, or a Values object.

- To view the message in XML format, select the **As XM**L link in the View field.

- To view the message in HTML format, select the **As HTML** link in the View field.
- To view the message as a Values object, select the **As Values** link in the View field.

➡ It is not possible to view non-IDocs in XML or HTML. They can only be viewed as Values object.

Narrowing the List of Transactions that Are Displayed

If you have several transactions and want to search for a specific transaction, you can edit your query settings. The Integration Server displays only the transactions that match the criteria that you specify.

You can only narrow the list of transactions if you use a JDBC-compliant database rather than the local file system for the Message Store.

To narrow the list of transactions that are displayed

1. Select the Routing task from the Integration Server navigation panel.
2. Select the **Transactions** tab if it is not already selected.
3. Choose **Edit Query Settings**. The Integration Server displays the Configure Transaction Queries screen.
4. Set the parameters on the screen as follows:

Key	Value
Sender	The name of the sender for the transactions you want to view. Use pattern-matching characters to select transactions that have similar senders. Specify '%' to view transactions from all senders.
Receiver	The name of the receiver for the transactions you want to view. Use pattern-matching characters to select transactions that have similar receivers. Specify '%' to view transactions from all receivers.
MsgType	The message type of the transactions you want to view. Use pattern-matching characters to select transactions that have similar message types. Specify '%' to view transactions for all types of messages.
State	The state of the transactions that you want to view. Select the state from the drop down list.

5. Choose View Transactions.

Deleting a Transaction from the Message Store

If the Message Store contains a message that you do not want the Partner Manager to process, you can delete it.

To delete a transaction from the Message Store

1. Select the Routing task from the Integration Server navigation panel.

2. Select the **Transactions** tab if it is not already selected.

3. Click ✕ in the **Delete** column for the transaction you want to delete.

Purging All Transactions from the Message Store

There are two procedures to periodically purge the message store. The following procedure, which is recommended, uses the wm.PartnerMgr.xtn.Sweeper:sweepTRX service to periodically purge the message store of transactions.

To purge all transactions from the message store periodically

1. In Developer, create a flow service that invokes the wm.PartnerMgr.xtn.Sweeper:sweepTRX service.

2. Pass the following parameters to the sweepTRX service.

For this parameter….	**Specify…**
onState	The transaction state. examples here next time
elapsedTime	The number of minutes that the transaction has been in **onState**
maxTrxCount	The maximum number of transactions to delete each time that sweepTRX is invoked

   When the sweepTRX service is invoked, the System finalizer is invoked and the garbage collector is invoked.

3. Schedule the flow service to run periodically to purge the message store.

Example

If you want to periodically purge all transactions which have been in the Confirmed state for longer than 30 minutes, you can create a service called 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 increasing the amount of overhead processing.

The following procedure deletes all transactions manually from the message store. Use this procedure only to delete a small number of documents. Otherwise, server performance can degrade substantially.

To purge all transactions from the message store manually

1. Start Server Administrator. See the *webMethods Integration Server Administrator's Guide* if you need procedures for this step.

2. On the **Adapters** menu, click Routing.

3. Click **Transactions**.

4. In the **Delete** column heading, click **Delete All**.

Chapter 7: Setting up Routing for BAPIs

Inbound Call to a SAP System

Setting Up the Integration Server and the BAPI Transport

Inbound messages in the new BAPI-based XML format can be easily dispatched using the Integration Server routing tool.

For each BAPI sent via XML to the Partner Manager, a corresponding entry in the routing table should exist. If no entry exists, the message won't be executed and the Integration Server will automatically generate a routing rule for this message. This routing rule has to be completed and activated by an administrator.

The message type used in the Partner Manager 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>).

After a routing rule has been automatically generated, it is not yet ready for processing. This is indicated by the red symbol in the **Status** column. By choosing **Edit** you can complete this routing rule.

The only transport applicable to inbound BAPI calls is the BAPI transport. This transport has two parameters:

- SAP Destination:. You can select a target SAP system from the drop down list. The message will be sent to this SAP system.

- 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 the documentation about the BizTalk Envelope, *BizTalkEnvelopeSpec.pdf*). If you only want to allow one specific type of call (for example for performance reasons only asynchronous calls, or for administration reasons only synchronous calls). You can define restrictions using the drop down list:

no restrictions	Caller might decide to send both synchronous and asynchronous calls
synchronous only	Caller might 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 might only send calls with a transaction ID. Messages without a transaction ID (synchronous messages) are rejected and an XML error message is returned

After completing the form, select *Save*. The routing rule is now displayed as *Ready*, indicated by the green symbol in the routing table.

Posting BAPI-Based XML to the Integration Server

To apply the routing rules, the XML message should be posted to the Partner Manager for BAPIs, this is the service pub.sap.transport.BAPI:InboundProcess.

http://*Integration Server host*:*Integration Server port*/invoke/pub.sap.transport.BAPI/InboundProcess

In the HTTP body, an XML message specifying the BAPI-call as described above must be sent.

You have to use the HTTP content type application/x-sap.busdoc when posting to the Integration Server.

```
POST /invoke/pub.sap.transport.BAPI/InboundProcess HTTP/1.0
Content-Type: application/x-sap.busdoc
User-Agent: Java1.1.8
Host: localhost:5555
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
Content-length: 817

<?xml version="1.0" encoding="iso-8859-1"?>
  <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
    <header>
      <delivery>
        <message>
          <messageID>0A125F1315B3D24B0000001E</messageID>
          <sent>2000-06-20T09:58:00</sent>
        </message>
        <to>
          <address>urn:sap-com:logical-system:SAPSYS0001</address>
        </to>
        <from>
          <address>urn:sap-com:logical-system:BUSCON0001</address>
        </from>
      </delivery>
    </header>
    <body>
      <doc:CompanyCode.GetList xmlns:doc="urn:sap-com:document:sap:business"
xmlns="">
        <CompanyCodeList>
          <item>
            <COMP_CODE></COMP_CODE>
            <COMP_NAME></COMP_NAME>
          </item>
        </CompanyCodeList>
      </doc:CompanyCode.GetList>
    </body>
  </biztalk_1>
```

Posting RFC Based IFR-compatible XML Messages

Integration Server 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 (see above), you can easily post RFC function calls to the Integration Server.

These XML messages must provide a BizTalk XML envelope (for further information, see the documentation about the BizTalk Envelope, BizTalkEnvelopeSpec.pdf). 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 the new 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://*host*:*port*/invoke/pub.sap.transport.RFC/InboundProcess.

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 Integration Server in some cases the SAP System expects a *commit* command to execute the call. To commit the transaction directly over HTTP do the following:

1. Call lockSession with empty POST over HTTP http://localhost:5555/invoke/sap/lockSession.

2. Call the BAPI in the same session using HTTP .

3. Commit the service with an empty POST using HTTP http://localhost:5555/invoke/sap.bapi.transaction/commit.

4. Release the session by sending an empty POST using HTTP http://localhost:5555/invoke/sp/releaseSession.

With the HTTP requests in steps 2-4 you need to resubmit the cookie that you got with the response in step 1. Otherwise the adapter cannot assign these requests to the same user session and the commit does not know which transaction to commit.

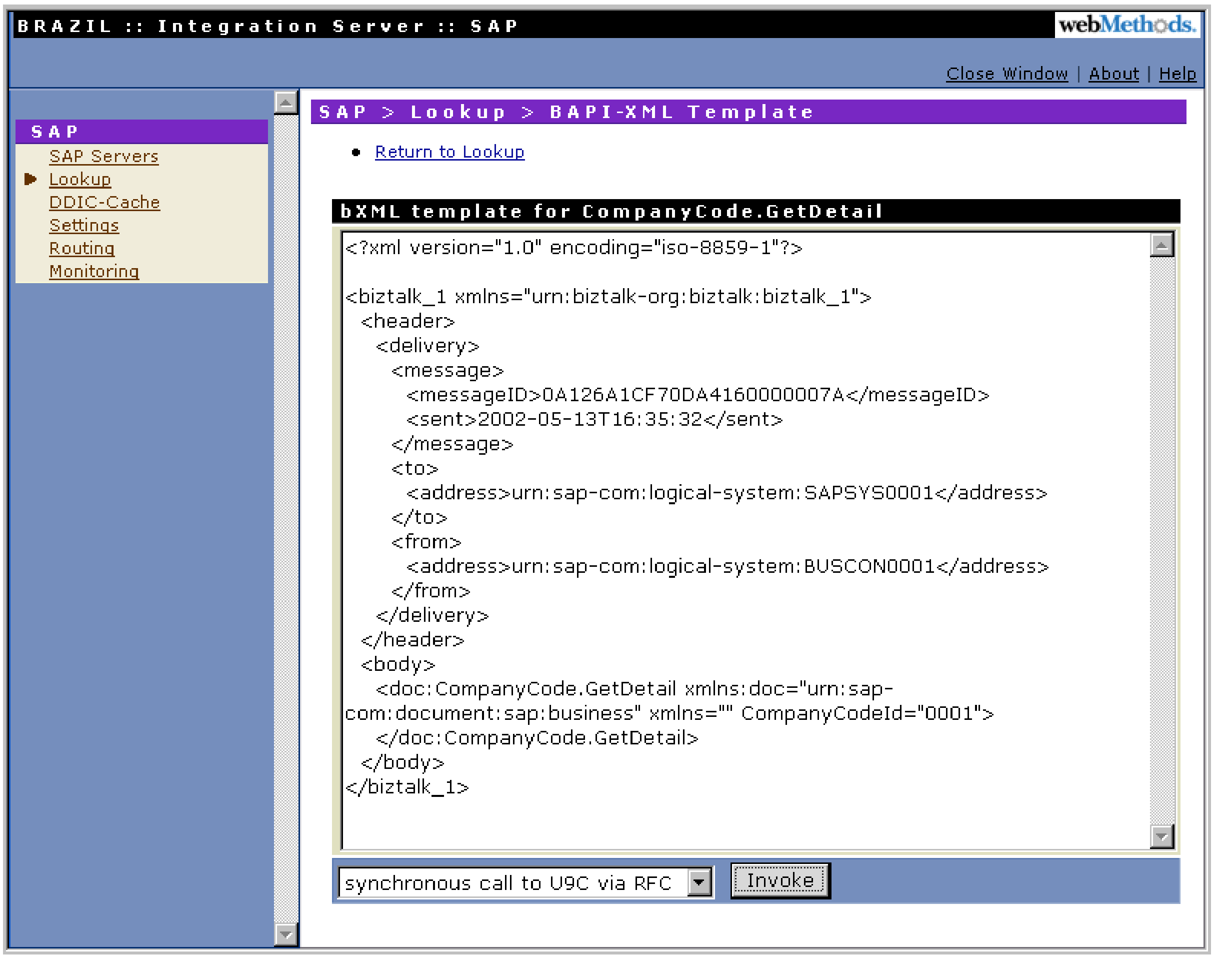
Synchronous Calls

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 FM.

Example

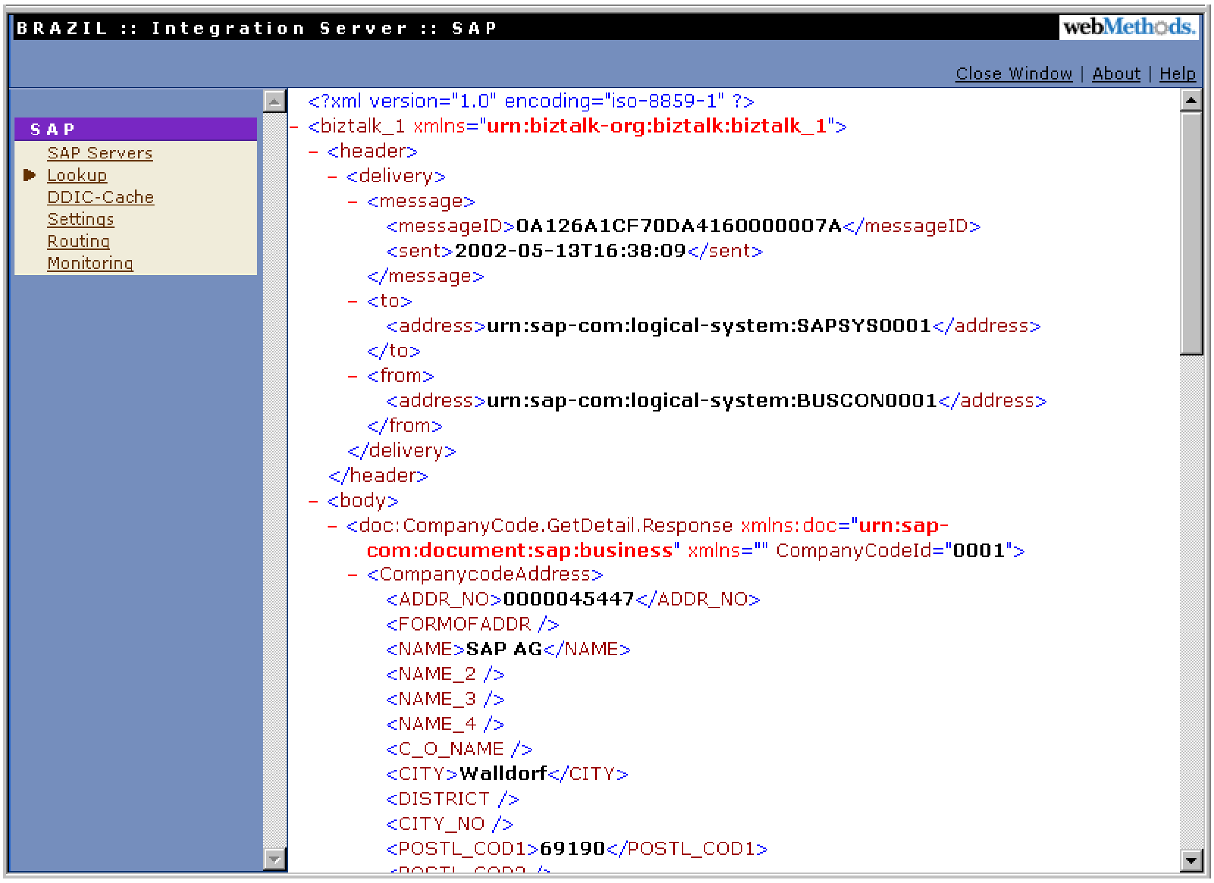
Sending a synchronous call to a SAP system:


BRAZIL :: Integration Server :: SAP
webMethods.
Close Window | About | Help
SAP
SAP Servers
Lookup
DDIC-Cache
Settings
Routing
Monitoring
SAP > Lookup > BAPI-XML Template
Return to Lookup
bXML template for CompanyCode.GetDetail
synchronous call to U9C via RFC
Invoke

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

<biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
  <header>
    <delivery>
      <message>
        <messageID>0A126A1CF70DA4160000007A</messageID>
        <sent>2002-05-13T16:35:32</sent>
      </message>
      <to>
        <address>urn:sap-com:logical-system:SAPSYS0001</address>
      </to>
      <from>
        <address>urn:sap-com:logical-system:BUSCON0001</address>
      </from>
    </delivery>
  </header>
  <body>
    <doc:CompanyCode.GetDetail xmlns:doc="urn:sap-
com:document:sap:business" xmlns="" CompanyCodeId="0001">
    </doc:CompanyCode.GetDetail>
  </body>
</biztalk_1>
```

The Integration Server returns a response business document:



```
  <?xml version="1.0" encoding="iso-8859-1" ?>
- <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
  - <header>
    - <delivery>
      - <message>
          <messageID>0A126A1CF70DA4160000007A</messageID>
          <sent>2002-05-13T16:38:09</sent>
        </message>
      - <to>
          <address>urn:sap-com:logical-system:SAPSYS0001</address>
        </to>
      - <from>
          <address>urn:sap-com:logical-system:BUSCON0001</address>
        </from>
      </delivery>
    </header>
  - <body>
    - <doc:CompanyCode.GetDetail.Response xmlns:doc="urn:sap-
        com:document:sap:business" xmlns="" CompanyCodeId="0001">
      - <CompanycodeAddress>
          <ADDR_NO>0000045447</ADDR_NO>
          <FORMOFADDR />
          <NAME>SAP AG</NAME>
          <NAME_2 />
          <NAME_3 />
          <NAME_4 />
          <C_O_NAME />
          <CITY>Walldorf</CITY>
          <DISTRICT />
          <CITY_NO />
          <POSTL_COD1>69190</POSTL_COD1>
          <POSTL_COD2 />
```

Asynchronous Calls to SAP Systems

An asynchronous call can be executed by specifying a transaction ID in the BizTalk header of the XML document. The XML element used is the `<referenceID>`-element (see description of the BizTalk XML envelope) in the `<receiver>`-section of the BizTalk header.

A transaction ID should be a global unique ID expressed in hexadecimal letters and 24 letters long, for example 0A11449F652C394A34DE042F.

If an asynchronous request can be transmitted without errors inside the SAP System, the Integration Server 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.

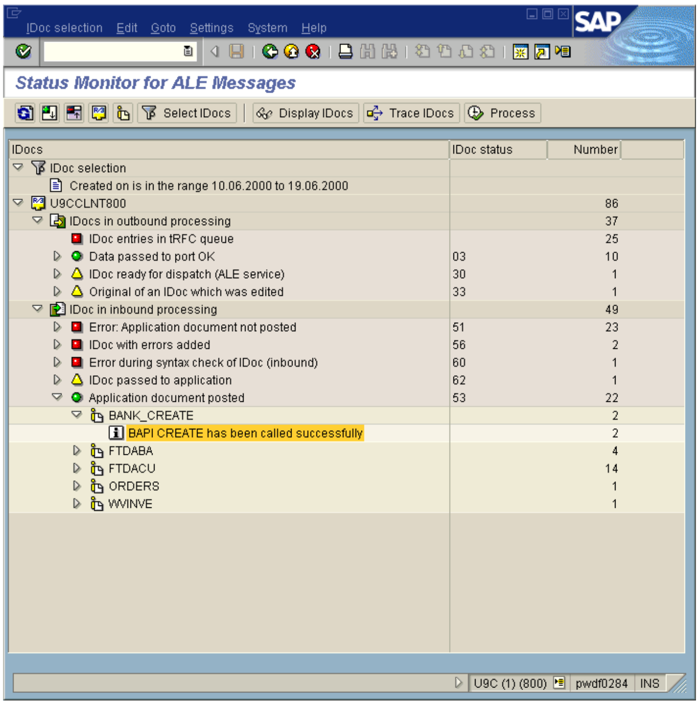
The result of an asynchronous call can be checked inside the target system by using the ALE monitoring services (transaction BD87) in a 4.6 SAP System. See your SAP system documentation for further information on ALE services.

Example

Sending an asynchronous request to a SAP System by specifying a transaction ID.

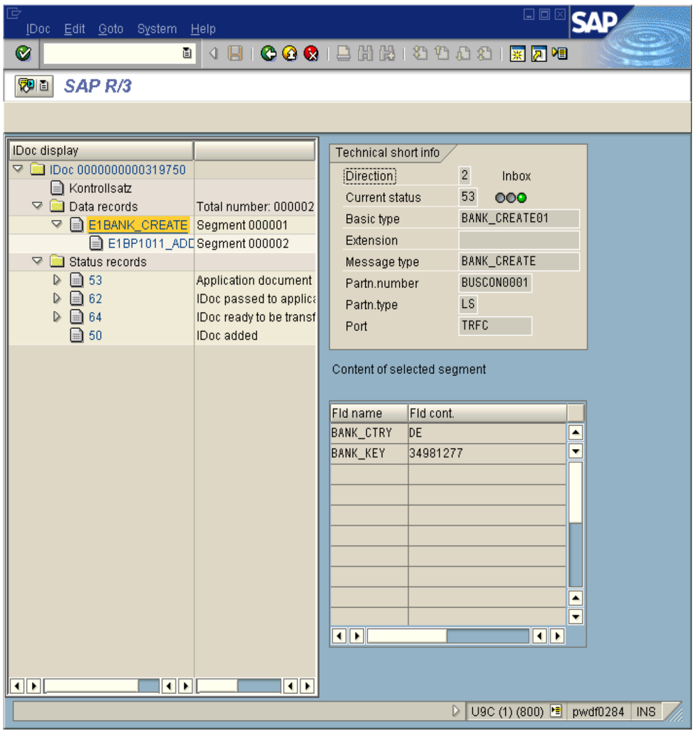
Receiving an XML confirmation document.

Checking the correct application-level processing inside the target SAP System (transaction BD87 in a 4.6 system or use the IDoc list transaction WE05):



Status Monitor for ALE Messages

IDocs	IDoc status	Number
IDoc selection		
Created on is in the range 10.06.2000 to 19.06.2000		
U9CCLNT800		86
IDocs in outbound processing		37
IDoc entries in tRFC queue		25
Data passed to port OK	03	10
IDoc ready for dispatch (ALE service)	30	1
Original of an IDoc which was edited	33	1
IDoc in inbound processing		49
Error: Application document not posted	51	23
IDoc with errors added	56	2
Error during syntax check of IDoc (inbound)	60	1
IDoc passed to application	62	1
Application document posted	53	22
BANK_CREATE		2
BAPI CREATE has been called successfully		2
FTDABA		4
FTDACU		14
ORDERS		1
WVINVE		1



IDoc display

IDoc 0000000000319750	
Kontrollsatz	
Data records	Total number: 000002
E1BANK_CREATE	Segment 000001
E1BP1011_ADI	Segment 000002
Status records	
53	Application document
62	IDoc passed to applic
64	IDoc ready to be transf
50	IDoc added

Technical short info

Direction	2	Inbox
Current status	53	○○●
Basic type	BANK_CREATE01	
Extension		
Message type	BANK_CREATE	
Partn.number	BUSCON0001	
Partn.type	LS	
Port	TRFC	

Content of selected segment

Fld name	Fld cont.
BANK_CTRY	DE
BANK_KEY	34981277

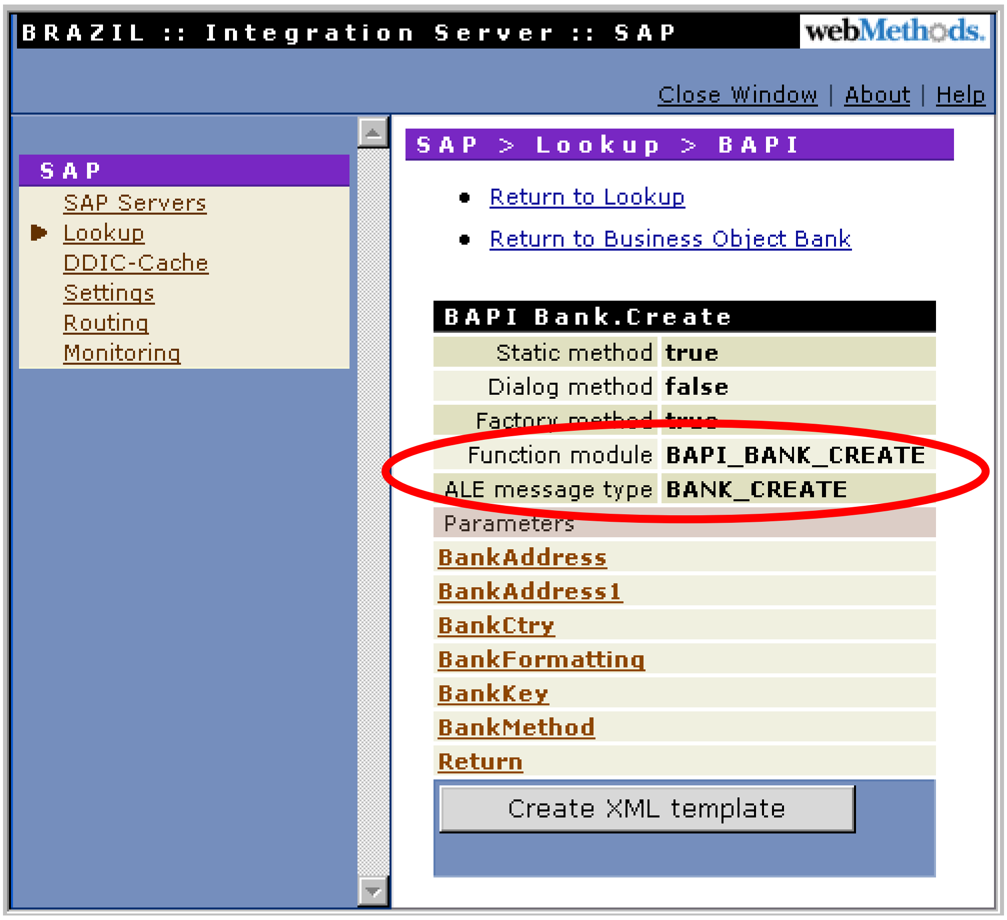
Outbound Call from a SAP System

Overview

SAP Systems communicate with the SAP adapter on an implementation-level when calling BAPIs. This means, they try to call the implementing function module directly via RFC or, if asynchronous processing is used, they send an IDoc.

The SAP adapter has a built-in converter to automatically rebuild the original object-based BAPI method call and represent it as an XML message.

Therefore, the XML transport can handle these messages using the XML dialect *bXML*, to build a BAPI-style XML message.



BAPI Bank.Create	
Static method	true
Dialog method	false
Factory method	true
Function module	BAPI_BANK_CREATE
ALE message type	BANK_CREATE
Parameters	
BankAddress	
BankAddress1	
BankCtry	
BankFormatting	
BankKey	
BankMethod	
Return	

To activate this function, you have to specify a routing rule from the SAP call to the XML transport.

The message types used in the Partner Manager for routing are based on the BAPI implementation. This means, for routing messages from an SAP Systems:

- 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 rule setup, you have to specify the name of the business object and BAPI to which the call should be mapped.

Setting Up Routing Rules for the XML Transport

To enable this function, you will first have to setup an RFC Listener for the relevant SAP Systems (see Chapter 5:).

You can prepare a routing-rule fragment by executing the corresponding call inside your SAP System with your adapter as the destination. If no routing rule has yet been specified, a new one will be created.

To edit this routing rule perform the following steps:

1. Select the **Routing** tab on the Integration Server Administration Web page.

2. Select **Edit** for the new routing rule (new routing rules are indicated by a red symbol in the **Status** column).

3. Select **XML** in the **Transport** field to choose the XML transport.

4. Select the XML Dialect **bXML**.

5. Enter a URL as destination of the XML call. To post the XML to another Integration Server, 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.

6. Select the checkbox **Use BAPI format** to specify that the call should be represented as a BAPI in XML.

7. Enter the name of the corresponding business object.

8. Enter the name of the corresponding BAPI method.

9. Optionally, you can specify a username and password for user authentification on the remote host.

With release 4.6, 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).

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, you only have to select *bXML* as XML dialect at the routing rules for RFC messages, and deselect the *Use BAPI format* option.

Dynamic Routing Using the XML Transport

As in previous releases of the SAP adapter, it is not always necessary to specify routing rules.

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 authentification on the remote web system
httpPassword (optional)	An optional parameter which allows you to specify a password for authentification on the remote web system
useBAPI	Set this value to ‘ON’ if you want to use the BAPI XML format. If you omit this value or set it to something different than ‘ON’, the message will be sent as RFC based XML in a BizTalk XML envelope
object	The name of the business object, to which the call should be mapped. This value has to be case-sensitive
bapi	The name of the BAPI method, to which the call should be mapped. This value is case-sensitive
xsender	A value that should be put in the header field <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 field <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 true (for SAP-XML and bXML).
useUTF8	Flag that allows to force the renderers to use the encoding utf-8, if set to true (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.
.
.
.
CALL FUNCTION 'BAPI_COMPANYCODE_GETLIST'
  DESTINATION 'SAPADAPTER'
```

```
IMPORTING
  RETURN                 = returnCode
TABLES
  COMPANYCODE_LIST       = companyCodes
  SBCHEADER              = header
```

Example: Calling a BAPI Synchronously from a SAP System

This example demonstrates how to invoke a call of the BAPI `CompanyCode.GetDetail` from an SAP System to the Web via XML.

As the BAPI will be sent to the Partner Manager, 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.

Preparations:

- Set up an RFC Listener for the SAP System on your Integration Server as described in Chapter 5:.
- Set up a corresponding RFC destination on your SAP System for your adapter using the transaction SM59 as described in Chapter 5:.
- Create a program ZBAPIDEMO in the SAP System using the transaction SE38

You can enter the following source code:

```
REPORT ZBAPIDEMO .
*variable companyCodes will take the application result
data companyCodes like BAPI0002_1 occurs 1 with header line.
*variable header gives adapter instructions for routing
data header like SBCCALLENV occurs 1 with header line.
*variable returnCode takes the processing information after the
*synchronous call
data returnCode type BAPIRETURN.
*Fill the SAP adapter routing instructions
header-name = 'sender'.
header-value = 'SAPSYS'.
append header.
header-name  = 'receiver'.
header-value = 'SAPADAPTER'.
append header.
*Execute the RFC with destination SAPADAPTER
CALL FUNCTION 'BAPI_COMPANYCODE_GETLIST'
*->SAPADAPTER should be maintained in SM59 as alias to the SAP adapter
  DESTINATION 'SAPADAPTER'
IMPORTING
  RETURN                 = returnCode
TABLES
  COMPANYCODE_LIST       = companyCodes
  SBCHEADER              = header[]
            .
*process the BAPI-Return parameter
if not returnCode is initial.
  write: / 'BAPI return parameter:'.
  write: / ' Code:    ' , returnCode-CODE.
  write: / ' Message: ' , returnCode-MESSAGE.
endif.
*display the application result
loop at companyCodes.
  write: / companyCodes-COMP_CODE, ' ', companyCodes-COMP_NAME.
endloop.
```

Before executing this report, you should specify a routing rule 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.

Prepare a routing rule for the RFC call:

1. Log on to your Integration Server administration web page.

2. Select the **Routing** tab on the Integration Server Administration Web page.

3. In the input fields at the bottom of the page, enter the following data:

   Sender: SAPSYS
   Receiver: SAPADAPTER
   Message-type: BAPI_COMPANYCODE_GETLIST

4. Click **Add rule**.

5. In the field **transport**, select **XML**.

6. 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

   If necessary, change port 5555 to your current settings.

7. Select XML dialect **bXML**.

8. Enable the **Use BAPI mode** checkbox.

9. Enter `CompanyCode` as business object.

10. Enter `GetList` as BAPI.

11. Click **Save**.

To handle the inbound XML message, which will now be sent back to the local Integration Server, you have to specify a second routing rule for the BAPI:

1. Log on to your Integration Server Administration UI.

2. Select the **Routing** tab on the Integration Server Administration Web page.

3. In the input field at the end of the page, enter the following data:

   Sender: SAPSYS
   Receiver: SAPADAPTER
   Message-type CompanyCode.GetList.

4. Click **Add rule**.

5. In the transport field, select **BAPI**.

6. Select the SAP server to which the message should be routed from the drop down list.

7. Click **Save**.

When executing the program in the SAP System, the SAP adapter will send the HTTP Request to the remote host, which will look similar to the following one:

```
POST /invoke/pub.sap.transport.BAPI/InboundProcess HTTP/1.0
User-Agent: Mozilla/4.0 [en] (WinNT; I)
Accept: image/gif, */*
Host: localhost:90
Content-type: application/x-sap.busdoc
Cookie: ssnid=553891555519
Content-length: 817

<?xml version="1.0" encoding="iso-8859-1"?>
  <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
    <header>
      <delivery>
        <message>
          <messageID>0A125F1315B3D24B0000001E</messageID>
          <sent>2000-06-20T09:58:00</sent>
        </message>
        <to>
          <address>urn:sap-com:logical-system:SAPADAPTER</address>
        </to>
        <from>
          <address>urn:sap-com:logical-system:SAPSYS</address>
        </from>
      </delivery>
    </header>
    <body>
      <doc:CompanyCode.GetList xmlns:doc="urn:sap-com: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 SAP adapter 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.

Example: Calling a BAPI Asynchronously from a SAP System

This example demonstrates how to call a BAPI asynchronously from an SAP System over the Web.

For this purpose, you have to create an ABAP program inside the SAP System. This program calls the proxy function module to generate an ALE message for a BAPI. After performing a *COMMIT WORK* command, the

SAP System will generate an IDoc through the ALE service layer and send it to the SAP adapter. There the IDoc will be transformed into the original BAPI representation and sent as XML using HTTP to any remote web system.

While BAPI function module names usually start with the prefix `BAPI_`, the corresponding ALE proxies are usually named equally with the prefix `ALE_`

To get to a running 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, you have to ensure, that the option Transfer IDoc immediately is selected, as the BAPI conversion tool on the SAP adapter does not support IDoc packages.

Create the following report in your SAP System to test the BAPI conversion with the BAPI *Bank.Create* (available as of SAP release 4.6C).

```
REPORT ZALEDEMO .
parameters receiver like BDI_LOGSYS-LOGSYS.
parameters country like BAPI1011_KEY-BANK_CTRY default 'DE'.
parameters bankkey like BAPI1011_KEY-BANK_KEY default '34981255'.

data: receivers like BDI_LOGSYS occurs 0 with header line,
      address like BAPI1011_ADDRESS.

*prepare the distribution information
move receiver to receivers-LOGSYS.
append receivers.

*prepare the BANK addres parameter
address-BANK_NAME = 'Demo Bank'.
address-REGION  = 'BW'.
address-STREET = 'Neurottstr. 16'.
address-CITY = 'Walldorf'.
address-SWIFT_CODE = 'ABCDDE12'.
address-BANK_GROUP = 'SB'.
address-BANK_NO = '12345678'.
address-ADDR_NO = '123'.

*send data to ALE
CALL FUNCTION 'ALE_BANK_CREATE'
  EXPORTING
    BANKCTRY                      = country
    BANKKEY                       = bankkey
    BANKADDRESS                   = address
  TABLES
    RECEIVERS                     = receivers
 EXCEPTIONS
   ERROR_CREATING_IDOCS          = 1
   OTHERS                        = 2
          .
IF SY-SUBRC <> 0.
  write: / 'IDoc could not be created'.
ELSE.
  write: / 'IDoc successfully created'.
ENDIF.
*trigger processing
commit work and wait.
```

To create similar routing rules

You can create routing rules similar to those described in the synchronous example.

1. Log on to your Integration Server Administration Web page.
2. Select the **Routing** tab..
3. In the input field at the end of the page, 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.
4. Click **Add rule**.
5. In the transport field, select **XML**.
6. Specify a URL for the target system. To test the functionality using your local SAP adapter's inbound processing, enter:

   http://localhost:5555/invoke/pub.sap.transport.BAPI/InboundProcess

   If necessary, change port 5555 to your current settings.
7. Select XML dialect **BizTalk**.
8. Enable the **Use BAPI mode** checkbox.
9. Enter `Bank` as business object.
10. Enter `Create` as BAPI.
11. Press the **Save** button.

To handle the inbound XML message

To handle the inbound XML message, which will now be sent back to the local Integration Server, you have to specify a second routing rule for the BAPI:

1. Log on to your Integration Server Administration Web page
2. Select the **Routing** tab on the Integration Server Administration Web page.
3. In the input field at the end of the page, 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.
4. Click **Add rule**.

5. In the field transport, select **BAPI**.

6. Select the SAP server to which the message should be routed from the drop down list.

7. Click **Save**.

When executing the report, you have to enter a `RECEIVER`. This should be the name of the logical system you have defined for your Integration Server.

When executing the report in the SAP System , the Integration Server will transmit the XML document via HTTP (names of logical systems depend on your system configuration). The XML would look like this:

```
<?xml version="1.0" encoding="iso-8859-1"?>
  <biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
    <header>
      <delivery>
        <message>
          <messageID>0A125F1315B3A11B00000035</messageID>
          <sent>2000-06-07T09:22:40</sent>
        </message>
        <to>
          <address>urn:sap-com:logical-system:SAPSYS0001</address>
        </to>
        <from>
          <address>urn:sap-com:logical-system:BUSCON0001</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 (transaction BD87 in SAP release 4.6C, IDoc lists can also be displayed with transaction WE05). If any application errors occurred, they are listed in the IDoc monitor.

Chapter 8: Routing IDocs and XML Messages to the Partner Manager

Introduction

For the Partner Manager to route IDocs and RFCs, you must send the IDocs and the RFCs to the Partner Manager. The following lists the typical ways that you can send IDocs and RFCs to the Partner Manager.

Method	Refer to page
Sending an RFC from a SAP System to the Integration Server	5-11
Sending IDocs with ALE from a SAP System to	8-2
Posting IDoc- and RFC-XML (XRFC) Documents	8-10
Posting arbitrary XML documents	8-7

When the Partner Manager receives the IDoc or RFC, it matches the sender, receiver, and message type associated with the IDoc or RFC to its routing rules. When it locates a matching routing rule, it invokes the routing rule to route the IDoc or RFC.

Sending IDocs with ALE from a SAP System to the Integration Server

The Partner Manager receives IDocs from an SAP system if the tRFC (INBOUND_IDOC_PROCESS or IDOC_INBOUND_ASYNCHRONOUS) has no inbound map associated with it. When the Partner Manager receives the IDoc, the sender, receiver, and message type, which is extracted from the IDoc itself, are transferred. The Partner Manager uses this information when determining where to route the IDoc.

For information about how to map the IDoc information into another format for use by another application, see *Mapping the Data in an IDoc* on page 8-17.

⚠ Before the SAP adapter can receive the IDoc, you must define an RFC listener for the Integration Server. For instructions, see *Configuring an RFC Listener in the Integration* Server on page 5-34.

To set up an SAP system to send IDocs to the Integration Server, use the SAP GUI to perform the following steps:

Step	Description
1	Create an RFC Destination on the SAP server
2	Define a logical port
3	Create a partner (also known as a *logical system*)
4	Create a partner profile

Step	Description
5	Create a distribution model for the partner and message type

Step 1 Create an RFC Destination on the SAP System

For instructions on how to create an RFC Destination, see *Creating a RFC Destination on a SAP System* on page 5-2.

Step 2 Define a logical port

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

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

1. Click the **Transactional RFC** tree item and click **Create**.

2. Click **New Entries** in the toolbar.

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

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

Step 3 Create a partner (also known as a *logical system*)

A logical subsystem manages one or more RFC Destinations.

You can create a partner (logical system) using transaction SPRO_ADMIN. Alternatively**,** use the following menu path to do this: Main screen -> **Tools** -> **AcceleratedSAP** -> **Customizing** -> **Project Management** .

1. Choose SAP Reference IMG.

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

3. Press the green checkmark beside Define Logical System.

4. Click **New Entry**.

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

Step 4 Create a partner profile

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

1. Select the **LS (logical system) partner** type in the tree view and press the **Create** button in the toolbar.
2. Enter the partner you created in step 3 in the partner field and save the partner profile.
3. Click **Insert entry** below the outbound parameter table control.
4. Enter the message type of the IDoc (for example, MATMAS).
5. Enter the logical receiver port you created before and enter the basic type of the IDoc, (for example, **MATMAS03**).
6. Save the outbound parameter.
7. Click **Insert entry** below the inbound parameter table control.
8. Enter the message type of the IDoc, (for example, **MATMAS**) and the process code, (for example, **MATM**).
9. Save the inbound parameter.

Step 5 Create a distribution model for the partner and message type

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

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

1. Open the Cross-Application Components folder, then the Distribution (ALE) folder, then the Distribution Customer Model folder in the tree view. Press the green hook beside Maintain customer distribution model directly.
2. Create a new model using **Model** → **Create**.
3. Add a message type to your model, enter the sender in the dialog box (for example, ALRCLNT000), enter the receiver (for example, your logical system), and the message type (for example, MATMAS).

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

1. In the Main screen, choose Tools → AcceleratedSAP → Customizing → Project Management.
2. Choose **SAP Reference IMG**.
3. Expand the following nodes: **Basis Components** → **Distribution (ALE)** → **Modelling and Implementing Business Processes** → **Maintain Customer Distribution Model**.
4. Press on the green hook beside Maintain Customer Distribution Model (transaction **BD64**).
5. Change into the edit mode.

6. Choose **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 choose Add message type.

9. In the dialog box, enter the sender (for example, ALRCLNT000), the receiver (for example, your logical system), and the message type (for example, MATMAS).

Using the ALE Monitoring Features via the SAP Adapter

Introduction

When an IDoc is sent from an SAP System to the adapter, the status of the IDoc (as displayed in transaction WE02) will always be "03, Data passed to port OK" (i.e. passed to the tRFC queue of the SAP system). However, this status doesn't provide any information regarding whether the IDoc could be processed successfully by the adapter, and who the final recipient is. To get this kind of information, the adapter offers three options: IDoc Trace, ALEAUD IDoc and SYSTAT01 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) → IS1 →Internet(http)→ IS2 → SAP System (receiver).

In all other cases status update via SYSTAT01 IDoc should be used. SAP adapter acts here like an EDI Subsystem and returns status information and information about the final receiver (receiving email address, Web Server URL, FTP Server, third party system, etc.) to the sender.

These three options are now described in detail:

IDoc Trace

Overview

The ALE Monitor allows you to make inquiries about the processing status of an IDoc in the receiving system actively from within the sending system. In R/3 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, and the system then makes a synchronous RFC call (IDOC_DATE_TIME_GET) to the LS that received the IDocs, with a list of DOCNUMs corresponding to the selected IDocs. (From transaction BD87 you have to hit the toolbar button "Trace IDocs" after the selection of the IDocs.) This function call returns for each IDoc the DOCNUM, under which it was saved in the receiving system, the receiving time and the current processing status in the receiving system.

Prerequisites

- The IDoc Trace functionality only works for IDocs exchanged between partners of Partner Type LS (Logical System).
- In order to be able to make this synchronous RFC, the partner profile of the LS, which represents the adapter, must contain an outbound parameter with message type SYNCH.
- Also the RFC to an LS of type “tRFC Port”, like the adapter, is only based on a system of release greater or equal 4.6D. 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

- You have to configure the adapter to use a Database as Message Store. See the *webMethods Integration Server Administrator’s Guide* for more information on this.

Prepare the Adapter for IDoc Trace

Proceed along the following steps to set up the adapter for IDoc tracing:

1. As described under *Configuring the Location of the Message Store* make the adapter use a database with $dbAlias “transactions” as Message Store
2. Shut down the adapter and add the following line to the file

   ```
   ...\Server\config\server.cnf:
   watt.sap.monitorIDocs=true
   ```
3. Start the adapter and then execute the Service "sap.monitor.idoc:startup".This will create a table SAPTRANSACTIONS in the database.

   If you don’t want the adapter to create tables in your database, you can instead create it manually as follows:

DOCNUM	VARCHAR 16
SOURCE_SYSTEM	VARCHAR 10
TID	VARCHAR 24
SENDER	VARCHAR 50
RECEIVER	VARCHAR 50
MSGTYPE	VARCHAR 50

4. For each SAP Server from which you want to receive and trace IDocs, create a Function Module Inbound Map for IDOC_DATE_TIME_GET and in it enter under “Outbound Handling”:

   ```
   Folder:   sap.monitor.idoc
   Service/Flow: idocTrace
   ```

5. All remote SAP adapters, to which IDocs are passed on via a remote invoke of pub.sap.transport.ALE:InboundProcess (or wm.PartnerMgr.gateway.transport.ALE:InboundProcess on olderadapterreleases), should 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 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 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

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 SAP adapter.

Further Setup in the SAP Adapter

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 SAP Adapter, who 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:

```
RCVPRN RCVPRN RCVPRN RCVPRN RCVPRN = T90CLNT090 RCVPRN = T90CLNT090
```

This can be achieved by adding the service sap.monitor.idoc.mappings:setReceiverToLS as preprocessing service to the Routing Rule and hard coding the corresponding values there.

Status Update via SYSTAT01 IDoc

Overview

In this case the SAP Adapter can be considered as a kind of EDI Subsystem. If the SYSTAT01 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 Rule, whether SYSTAT01 information should be reported to the sender or not. The following information is then reported in addition to the status:

- Program that set the status (SAP adapter on Integration Server)
- Routing Rule, which processed the IDoc
- The key (TID) under which the IDoc can be found in the adapter
- Date and time of status change
- In case of an error status: explicit error message
- In case of a success status: explicit information about the final receiver in the following form:
  - ALE Transport: IDoc submitted to <SAPSystem> with TID <TID>
  - XML Transport: Sent Msg to <URL>
  - Email Transport: Sent Msg to <emailAddress>
  - FTP Transport: Sent Msg to <hostname>:<port>
  - B2B Transport: Sent Msg to service: <folderPath>:<serviceName>

The adapter sends the following status:

- If everything went ok the first time: "06 Translation OK" and "12 Dispatch OK"
- If an error has occurred the first time: "11 Error during dispatch"
- If everything went ok at a later retrial: "13 Retransmission OK"
- If there was still an error at a later retrial: "23 Error during retransmission"
- If no information on the IDoc could be determined: "04 Error within control information of EDI subsystem"

Set Up the Participating SAP Systems

In each SAP System that is to receive SYSTAT01 IDocs from the adapter, 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:

1. Go to transaction 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.

2. Reassign the process code "STA1" from "without ALE service" to "with ALE service," if this has not been done yet. For 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 the node Inbound with ALE service → Processing by task

    4. In the following screen press the **Save** button (F11).

General setup:

1. In transaction SALE (Distribution (ALE)) go to **Basic configuration** → Set up logical system → Maintain logical systems" and add a new logical system for the adapter, if you don't have one yet. If you name it "BUSCON," the adapter will recognize it automatically, else you will have to make the name known to the adapter as described later.

2. In transaction WE20 (Partner Profiles) create a partner profile with "Partn.number" = the name you chose in step 1 (for example, BUSCON) and "Partn.type" = "LS". After you saved it, add an Inbound Parameter to it as follows:

    "Message type" = "STATUS"
    "Process code" = "STA1"

Prepare the Integration Server for Automatic SYSTAT01 IDoc

In the Integration Server you have to set up these elements to enable automatic status update via SYSTAT01:

1. If you chose a different name from "BUSCON" for the logical system above, shut down the Integration Server, add the following parameter to the server.cnf file in the *Integration Server_directory*\config directory, then restart the server:

    watt.sap.systat01.partnerNumber=<NameOfLogicalSystem>.

    With "BUSCON" you can omit this step.

2. Go to **Adapters** → **Routing** → **Routing Rules** and add the following Service in the field **Pre-Processing Service** for each Routing Rule, for which you want to enable SYSTAT01 Status update (or include it as the first step of the Pre-Processing Service, if you have already your own Pre-Processing Service defined):

    pub.sap.monitor.systat01:enable

    All IDocs, which come in via this Routing Rule, are now marked for automatic status update.

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 SYSTAT01 IDocs from getting too big, you should schedule it often enough, so that at most around 2000 IDocs have been received by the SAP adapter since the last run of the report Service.

   This service collects all the necessary information and error/success messages for those IDocs, which the Routing Rules chosen in step 2 have processed since the last run. For each SAP system that has sent any such IDocs to the SAP adapter, it creates and submits one SYSTAT01 IDoc containing all this information.

Posting IDoc- and RFC-XML (XRFC) Documents

To post an XML document into the Partner Manager, 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- and RFC-XML documents.

Posting an IDoc from a Web Browser

You can submit e.g. ORDERS and ORDRSP IDocs to the Partner Manager or to an SAP server from a Web browser using either of the following page:

http://*Integration Server host*:*port*/SAP/Submit_IDocXML.html

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
Content-Type: application/x-sap.idoc
Content-length: 1578

<?>xml version="1.0"?>

<SYIDoc01>
...
</SYIDoc01>
```

➡ The value for the content length has to be replaced by the actual length of content.

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

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

2. Choose **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 Arbitrary XML Documents

Starting with SR 2 for SAP adapter 4.0, it is possible to use XML Inbound Process to forward arbitrary XML documents to the PartnerManager 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…
boundNode	This record contains the XML document as it would look like after having been processed by pub.web:documentToRecord. Your service should extract the routing data from your specific documents.

Return Values

This key	Must specify…
sender	Sender used for finding the matching Routing Rule in the Partner Manager.
receiver	Receiver used for finding the matching Routing Rule in the Partner Manager.
msgType	Message type used for finding the matching Routing Rule in the Partner Manager.
$tid (optional)	Transaction ID found in the document, if client wants to execute a transaction once and only once. If no $tid exists, the Partner Manager will generate one.
$routeOnly (optional)	If this parameter is set to true, the PartnerManager, will only route the message, without creating a transaction.

To configure the Inbound Process mechanism of the XML transport, perform the following steps:

1. Create an inbound routing info service for your XML transport. It extracts all required parameters to do a routing with the Partner Manger

2. After having implemented a service, you still have to make it known to the XML transport. Go to "Routing ..." in the main Administrator screen of the Integration Server.

3. In the Partner Manager screen click on Transports, which will give you a list of all Transports:

4. Click on XML, and the configuration screen is shown to you. Provide the full service name in the input field, and push the save button:

Now you're ready to do routing using XML Inbound Process for arbitrary XML documents.

Mapping IDocs to Other Formats

If you need to use the information contained in an IDoc in documents of another format, you build a Flow 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 infor-mation from the IDoc fields to fields within the EDI system’s purchase-order record. Then, you define a routing rule that triggers this flow service.

The following instructions will show you how to do this in detail:

Stage	Description
1	Create the routing rule that routes the IDoc to the service that maps the IDoc information.
2	Create an empty flow service, which will later execute the mapping.
3	Create a Record Definition for your IDoc
4	Transform the IDoc to hierarchical format (i.e., transform it to a *boundNode* variable).
5	Map the IDoc fields to other variables in the pipeline.
6	Test the Mapping Service.
7	Remove Development Flow Operations from the Service

Creating a Routing Rule that Points to a Flow Service

Stage 1 Create the Routing Rule

The first step in building an application that maps information from an IDoc is to create a routing rule that invokes the flow service that will perform the mapping.

When you create the routing rule, select the B2B Service transport and specify the Configure B2B Service parameters to identify the location of the service that maps the IDoc. (You will create this service in a subsequent step.)

Stage 2 Create an Empty Flow Service

After you have created the routing rule, use the following procedure to create a flow service with Developer (this is the service that the Partner Manager will invoke based on the routing rule created in the previous stage).

To create an empty flow service

1. Open Developer and connect to the Integration Server that you identified in the routing rule you created in Stage 1.

2. On the *File* menu, click *New* and create an empty flow service. Make sure that you give the service the same folder and service name specified in the routing rule you created in Stage 1.

Obtaining a Record Definition for Your IDoc

Regardless of the way in which you pass the IDoc to the flow, you need to create a record describing the content and structure of the IDoc. There are two ways to create this record:

- You can generate it from the DTD for the IDoc.

- You can generate it from a "sample" IDoc that you capture with Developer at design time.

The first option produces the most complete Record, since it is based on the IDoc's DTD, which describes all possible fields in the IDoc. We recommend that you use this option whenever possible. Use the second option only if a DTD is not available for the IDoc with which you are working. Because this option derives a record from a sample document, there is no assurance that the Record will include all possible fields. Fields that aren't used in the sample document, won't appear in the record definition.

Generating a Record Definition from a DTD

Stage 3 Create a Record Definition for your IDoc

If you are using SAP R/3 version 4.6A or higher, you can create a DTD for an IDoc from transaction WE60. (See your SAP R/3 user guide for procedures). If you want to generate a record definition 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 record definition.

To create a record definition from an IDoc DTD

1. Open Developer.

2. On the **File** menu, select **New**.

3. On the New panel, select **Record** and then choose **Next**.

4. On the Record panel, do the following:

   In the **Folder** tree, select the Folder into which you want to save the record definition.

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

6. Select **Next**.

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

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

9. Choose **Finish**.

10. If your IDoc is earlier than Version 3, edit the record definition to remove the “40” designation from the control header element. For example, change **EDI_DC40** to **EDI_DC**.

11. Select 💾 to save the Record.

Generating a Record Definition without a DTD

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

To create a record definition without a DTD

1. Open Developer.

2. On the **File** menu, select **New**.

3. On the New panel, select **Record** and then choose **Next**.

4. On the Record panel, do the following:

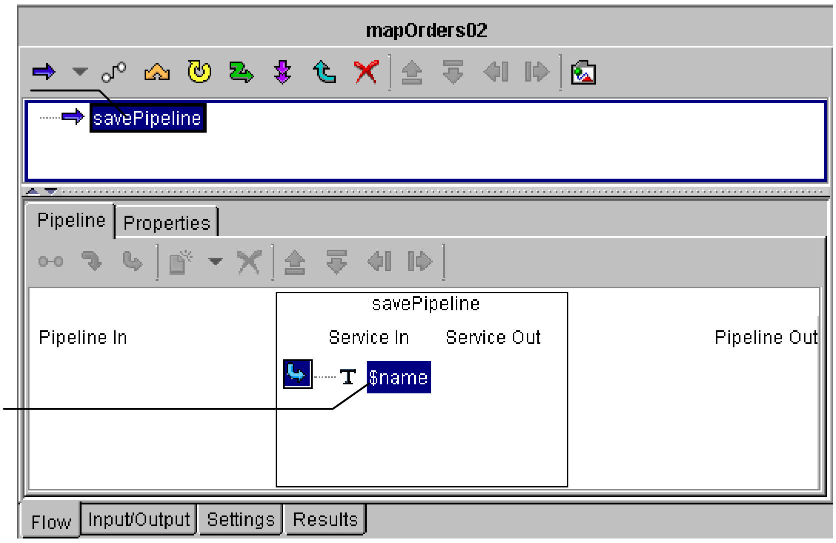
   In the **Folder** tree, select the Folder into which you want to save the record definition.

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

5. Press **Finish**.

6. Now switch to the empty Flow Service (projectA.idocs:mapOrders02) you created in stage 2.

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



Add the savePipeline service to the flow…

…and set this variable to assign a name to the saved pipeline

8. Select the **Pipeline** tab.

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

10. Type a name for the saved pipeline (for example, **SaveIDoc**) and select **OK**.

11. Choose to save the flow service.

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

    1. Use the SAP GUI or the utility at /SAP/Submit_ IDocXML.html to submit your sample IDoc to the Partner Manager. Specify the sender, receiver, and message type that you used for the routing rule that you created in Stage 1.

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

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

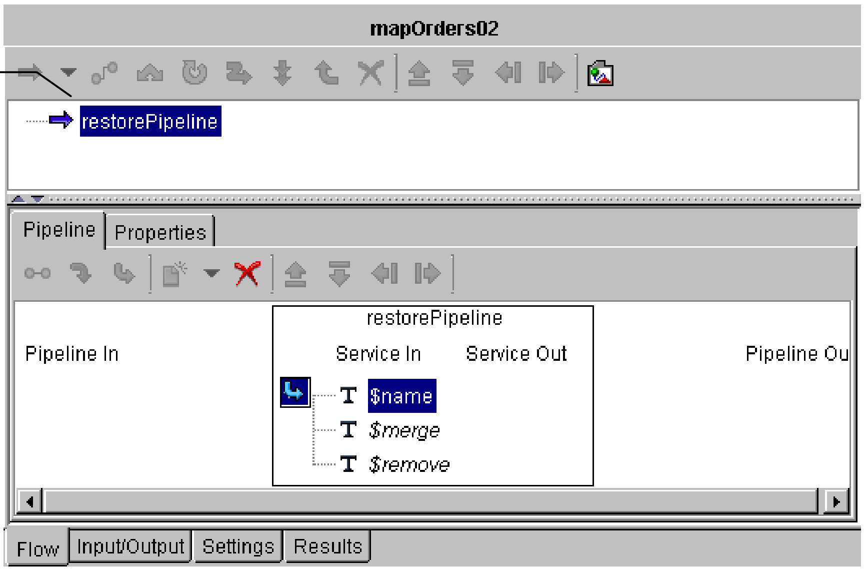
Use the following procedure to retrieve the pipeline image you created:

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

2. Select the **Flow** tab.

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

At this point, your flow contains only the restorePipeline



4. Under **Service In**, select the $name variable under and choose ↪ on toolbar in the **Pipeline** tab.

5. Specify the name of the saved pipeline and select **OK**.

6. As the next step of this service insert *pub.web:documentToRecord* (to be found also in the WmPublic Package), if the IDoc was sent to the adapter via http, or pub.sap.idoc:transformFlatToHierarchy (to be found in the SAP Package), if the IDoc was sent to the adapter from an SAP system via tRFC or from the demo page /SAP/Submit_ IDocXML.html.

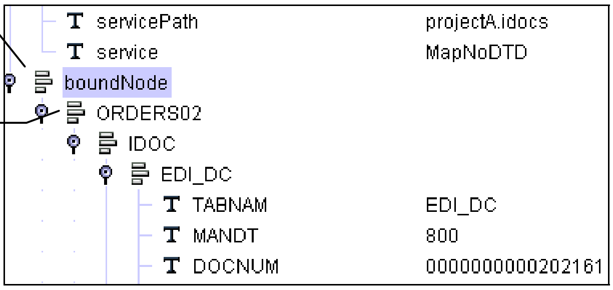
7. Choose 💾 to save this flow service.

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

   Now we can continue creating our Record Definition:

9. Select the root Record defined within *boundNode*. In the following example, the root Record is ORDERS02.

Locate the boundNode variable…



```
T servicePath          projectA.idocs
T service              MapNoDTD
boundNode
  ORDERS02
    IDOC
      EDI_DC
        T TABNAM       EDI_DC
        T MANDT        800
        T DOCNUM       0000000000202161
```

10. Select **Edit ▸ Copy** to make a copy of the root Record**.**

11. Now switch back to the empty record definition you created in step 4 and mark the empty pane on the right hand side. Select **Edit ▸ Paste** to paste the structure of the captured IDoc into your Record Definition.

12. Choose to save this Record Definition.

Mapping the Data in an IDoc

This section describes how to build a flow service that maps IDoc information into another Record structure. To accomplish this, follow the procedures in the stages below.

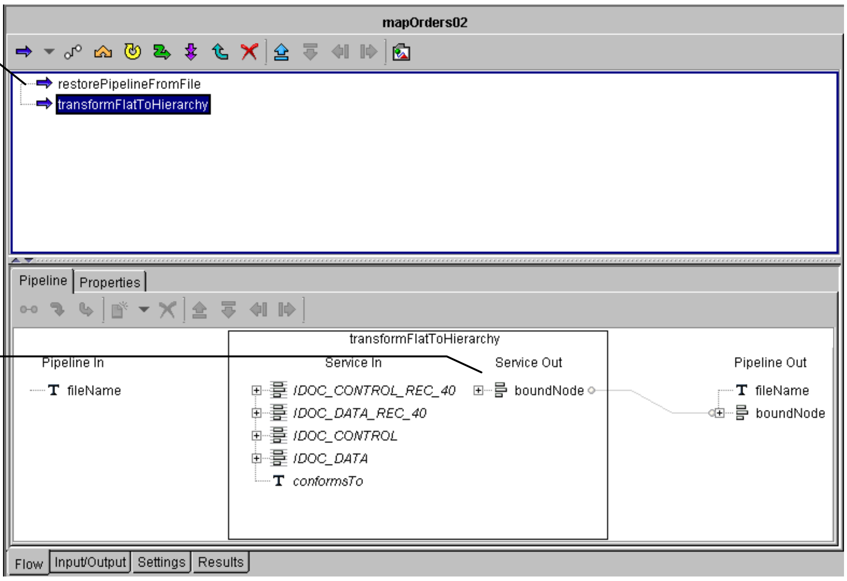
The new mapping service could be used as a pre-processing flow in a routing rule which handles such IDocs and should forward a document created in the mapping.

Stage 4 Transforming the IDoc to Hierarchical Format

To extract information from an IDoc, your flow service must transform the IDoc into a hierarchical structure and generate a boundNode—a record whose elements correspond to the fields in the IDoc. To do this, either use the transformFlatToHierarchy service (if the IDoc will be received by the adapter via tRFC or via http POST with Content-Type: application/x-sap.idoc) or the *documentToRecord* service (if the IDoc will be received via http Post with Content-Type: text/xml).

To transform the IDoc to hierarchical format

1. Select on the **Flow Pane** toolbar and choose the pub.sap.idoc:transformFlatToHierarchy service or the pub.web:documentToRecord service. (If this service does not appear in the list, select **Browse…** to locate it.) Either service transforms the IDoc into a boundNode variable, which contains the contents of the IDoc in a format that you can map to other pipeline variables.



You add the transformFlatToHierarchy service to your flow..

…to produce a boundNode

The picture above includes a restorePipeline operation. This is for testing and debugging the mapping as described later in stage 6. In production, your flow will not contain this flow operation.

2. Select to save this flow service.

Stage 5 Mapping IDoc Information to Pipeline Variables

After transforming the IDoc into a *boundNode*, 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 record definitions 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 record describing the content and structure of the IDoc.

1. In Developer, select the **Flow** tab.

2. In the Flow Pane, select the transformFlatToHierarchical/documentToRecord operation.

3. Select the **Pipeline** tab and choose the boundNode variable under **Pipeline Out**.

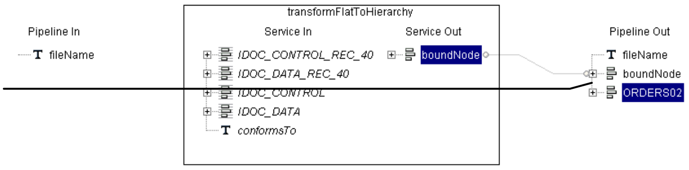
4. Choose on the toolbar and select **Record Reference**.

5. In the **Name** field, type the fully-qualified name of the record definition, you created in stage 3, or select it in the Folder tree.

6. Press **OK**.

7. Type a name for this Record and press ENTER. (You can give this Record 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 record that defines the structure of your IDoc. (Don't save the flow at this stage, because if you do, the yet unused *Record Reference* will be deleted again from the service...)

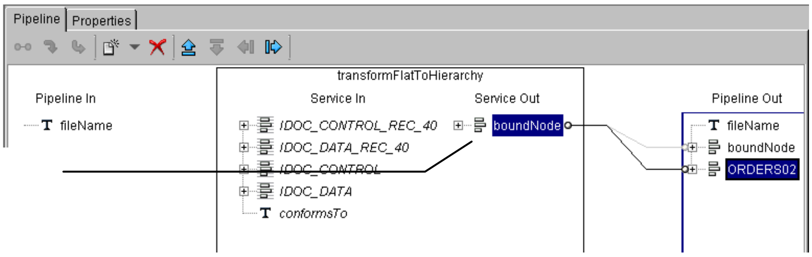


Insert the record definition in *Pipeline Out*

8. Take the following steps to map the boundNode variable to the record you just created.

9. In **Service Out**, select the boundNode variable.

10. In **Pipeline Out**, select the IDoc Record and press **Map** , or map boundNode to the IDoc Record using drag and relate. The Pipeline Editor will show a connecting line between boundNode and the IDoc Record you created in the previous procedure.



Map the boundNode variable to the record definition

Perform the following steps to define the variables to which you want to map information from the IDoc.

1. Choose on the **Flow Pane** toolbar.

Make sure the MAP operation appears immediately after the pub.sap.idoc:transformFlatToHierarchy operation. If necessary, use the arrow buttons on the toolbar to move it to the correct position.

2. Select the **Pipeline** tab.

3. In **Pipeline Out**, select the bottom most variable, and then

4. Choose from the toolbar and select the type of variable to which you want to map an IDoc value.

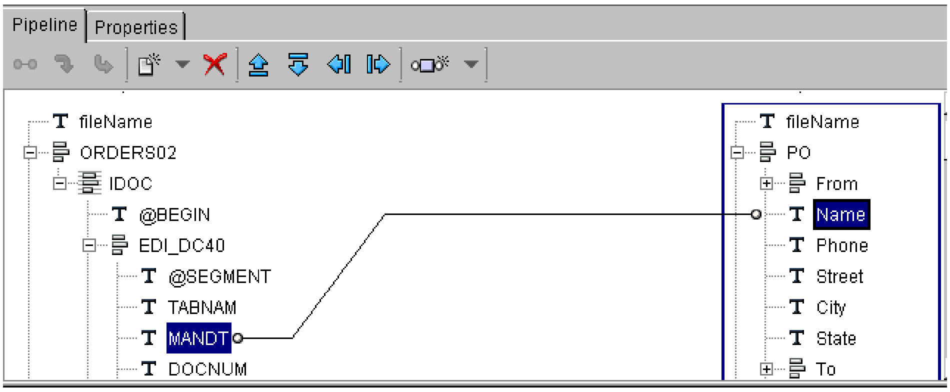
5. Type the name of the variable and press **ENTER**.

6. If the variable is a Record or a Record List, repeat steps 1 and 2 to define its member variables. Then use to indent each member variable beneath the Record or Record List variable.

7. Repeat this set of steps for each variable to which you want to map an IDoc value. For additional information about creating variables, see ”Adding Variables with the Pipeline Editor” in the *webMethods Developer User’s Guide*.

8. Use to map fields from the IDoc Record in **Pipeline In** to the appropriate target variables in **Pipeline Out**. If you need additional information about this step, see “Mapping Pipeline Variables to Service Inputs and Outputs” and “Working with the MAP Operation” in your *webMethods Developer User’s Guide*.

   The following example shows a variable mapped from the ORDERS02 IDoc to a String variable in Record called PurchaseOrder.



You can find a record definition for the example purchase order shown above in sap.demo.records:PurchaseOrder.

Stage 6 Test the mapping service

To test the flow service you created, use the SAP GUI or the /SAP/Submit_ IDocXML.html utility to submit your sample IDoc to the Partner Manager. Make sure to specify the sender, receiver, and message type that you used for the routing rule that you created in Stage 1.

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

If you want to debug your service in Developer, you can use the savePipelineToFile and restorePipelineFromFile services to capture an IDoc submitted by the Partner Manager. To use this technique, do the following.

Use the following procedure to add the savePipelineToFile service to the beginning of your flow service:

1. Open your flow service in Developer.

2. Select ➡ ▾ in the **Flow Pane** toolbar, and select the pub.flow:savePipelineToFile service. If this service does not appear in the list, select **Browse…** to locate it.)

Make sure savePipelineToFile is the first operation in your flow. If necessary, use the arrow buttons on the toolbar to move it to the correct position.

3. Select the **Pipeline** tab.

4. Select the fileName variable under **Service In** and choose ↳ on the toolbar.

5. Type a name for the saved pipeline information (for example, SaveIDoc) and press **OK**.

6. Select 💾 to save this flow service.

7. Use the SAP GUI or the /SAP/Submit_ IDocXML.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.

Use the following procedure to delete the savePipelineToFile service and insert the restorePipelineFromFile service.

1. Open your flow service in Developer.

2. Select the **Flow** tab.

3. Select the pub.flow:savePipelineToFile operation and choose ✖ to delete it.

4. Choose ⇨ ▾ in the **Flow Pane** toolbar, and select the pub.flow:restorePipelineFromFile service. If this service does not appear in the list, select **Browse…** to locate it.)

5. Use the ⇧ to move the restorePipelineFromFile service to the top of the flow.

6. Select the **Pipeline** tab.

7. Select the fileName variable under **Service In** and choose ↳ on the toolbar.

8. Type a name of the saved pipeline information and click **OK**.

9. Select 💾 to save this flow service.

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

When you are finished testing, use the following procedure to delete the restorePipelineFromFile service and save the finished flow.

11. Select the **Flow** tab.

12. Select the pub.flow:restorePipeline operation and choose ✖ to delete it.

13. Choose 💾 to save your flow service.

Pre-Routing Mechanisms for Content-Based Routing with IDocs

It is sometimes necessary to do some manipulation on IDocs before processing them in the adapter or before sending them to an SAP server. This is most likely done when there are customized IDocs which need specific processing. The two most common manipulations are 'Content based routing' and 'mapping.' Both mechanisms can be performed on IDocs routed to the adapter. For IDocs going to an SAP server, only mapping manipulation applies.

'Content based routing' allows you to specify routing parameters (sender, receiver and msgType) overriding parameters in the header of the incoming IDocs. You determine such headers from the content of the IDoc itself. E.g. if 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': the mechanism that adds (or removes) data segments to (from) a document. E.g. 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 which contains its receiver and sender information in specific fields in the IDoc itself. Further, it contains extra fields that you want removed, or added, before you process the IDoc as an ORDER02 IDoc.

In the releases lower than 4.0, you had to modify existing flowservices with Developer and insert branches for the dedicated message types to achieve content based routing and mapping. Those settings have been overwritten when you upgraded the adapter.

wm.PartnerMgr.gateway.transport:performOutboundMapping
wm.PartnerMgr.gateway.transport:performInboundMapping

The services wm.PartnerMgr.gateway.transport:performOutboundMapping and wm.PartnerMgr.gateway.transport:performInboundMapping do not exist any more as of release 4.0. An alternative mechanism is described below.

The two services where you put all the logic for detecting and performing IDoc manipulations are now replaced with a registration mechanism. The registration mechanism assures that existing content-based routing settings are not overridden when upgrading the Integration Server.

You now register special pre-routing services based on message types. Also, a standard processing service can be established which will 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 (sent to the Business Connector) using pub.sap.transport.ALE:InboundProcess.

The outbound services are executed before an IDoc is sent to an SAP system by pub.sap.transport.ALE:OutboundProcess.

There are two services available, with which routing services can be registered and unregistered:

- pub.sap.idoc.routing:registerService: This service registers a given service with the routing manager. The best way to use it is to invoke it in a replication or startup service.

Input	**Description**
msgType	The message type of the IDoc for which you want to register a content based routing or mapping. If the special value $default is used, a standard service will be registered.
service	The complete namespace reference of the service to register (format: folder.subfolder:servicename)
inbound	Flag that indicates whether to register an inbound (default) or an outbound service. Set to true or false.
Output	**Description**
activated	Flag that indicates whether the service could be activated as routing service. Set to true or false.
activeService	Only present if activated==false. In this case it contains the service that is currently active or $none if all registered services are deactivated.

- pub.sap.idoc.routing:unregisterService: This service unregisters a given service from the routing manager.

Input	**Description**
msgType	The message type of the IDoc for which you want to unregister a content based routing or mapping. If the special value $default is used, a standard service will be unregistered.
service	The fully specified service to unregister
inbound	Flag that indicates whether to unregister an inbound (default) or an outbound service.
Output	Description
wasActive	Flag that indicates whether the unregistered service was the active routing service.

The best place to invoke those services for your routing service is within a replication service or a startup service for a package. Which of these services you choose is depending on your own needs. Note, that also the default routing service can be manipulated using the special message type *$default*.

A routing service has to use the following specifications for proper behaviour:

Specification	Description
pub.sap.idoc.routing:inbound	Specification that should be used for services which provide an inbound content based routing/mapping.
pub.sap.idoc.routing:inboundDefault	Specification that should be used for services which provide a default inbound routing/mapping
pub.sap.idoc.routing:outbound	Specification that should be used for services which provide an outbound content based mapping.
pub.sap.idoc.routing:outboundDefault	Specification that should be used for services which provide a default outbound mapping.

Once you registered your routing services you can associate them with message types via the Administrator UI (SAP -> Routing *)* :

In the SAP Routing screen of the Administrator UI there are two sections to select services: a message type independent section (default handling section) and a message type dependent section.

A default handling service (inbound or outbound) is executed for all message types. If there are services selected in the message type dependent section, they will be executed for the corresponding message type in addition to the default services.

If the special value $none is selected, all registered services are set to inactive.

You must register the services for content based routing before they can be selected via the Administrator UI.

In case of creating your own default inbound routing service, the parameters ‘sender’, ‘receiver’ and ‘msgType’ have to be set by yourself. Those parameters are part of the specification. If not, you might get an error message.

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 service has been registered with '$default' message type then you will not have a choice for selecting one at all, which was the case with the adapter out of the box. '$none' choice is only there, if one or more services are registered.

The settings in the image above indicate that for all incoming IDocs with an ORDERS message type, the 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 SAP with ORDRSP message type, the sap.idoc.mappings:ordrsp service will be executed. This service can alter the IDoc by adding or removing fields.

Improving the Performance of the Partner Manager

The Partner Manager now offers four switches to improve throughput and performance. These switches control the way how transactions and messages are persisted. In order to be effective, these switches have to be added as statements to the server.cnf configuration file. Please read the Appendix B Server Configuration on how to set and modify the statements.

Chapter 9: Coding Client Applications and Services

Overview

Integration Server 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 server, and to receive an IDoc from an SAP server. For descriptions of the available built-in services, see*: SAP Adapter* API on page 1.

This chapter also shows how to use the IDoc Java API to construct an IDoc. For information about the IDoc Java API, see *Integration Server_directory*\packages\SAP\doc\api\index.html.

Invoking RFCs from Integration Server

You can use built-in services that are shipped with the Integration Server to invoke an RFC on an SAP server.

Calling SAP Services from Java Services

This section shows an example of how to invoke an RFC from a Integration Server Java service. The sample Java *TestBAPI.java* that is shown below logs on to an SAP server and invokes a BAPI.

For more information about developing services, see the *webMethods Developers Guide* and the online API documentation installed with the Integration Server (*Integration Server_directory\doc\apiJava\index.html*).

To call a SAP service from Java

1. Start Developer if it is not already running.

2. Under **Edit ➠ Preferences ➠ Services**, choose the options **Use old Java service signature**.

3. Create a Java service named TestBapi and enter the following source:

```
Values bapiArgs = new Values()
bapiArgs.put(“serverName”,”HS1”);
bapiArgs.put(“COMPANYID”,”000001”);
bapiArgs.put(“$rfcname”, “BAPI_COMPANY_GETDETAIL”);
try
{
     out = Service.doInvoke(“pub.sap.client”, “invoke”, bapiArgs);
}
catch (Exception e)
{
     return Service.throwError(e);
}
```

   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 SAP Server HS1.

Change the SAP Server alias to your own alias if necessary.

Receiving IDocs from a SAP System

To receive an IDoc from an SAP system and pass it to an Integration Server service, route the IDoc through the Partner Manager. To route an IDoc through the Partner Manager to a service, you need to set up a routing rule for the IDoc. For instructions on establishing routing rules, see *Establishing Routing Rules* on page 6-9. When choosing the transport for the routing rule, select the B2B service transport. After selecting B2B service, you need to supply information that indicates to which service you want the SAP adapter to pass the IDoc.

The SAP adapter passes the IDoc to your Integration Server service as a Values object. You can use the IDoc Java API to access and modify the information in the IDoc.

Accessing and Modifying Fields in IDocs

The following code sample shows how to use the IDoc Java API 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.

```
Values idoc_pipeline = in;
IDoc idoc = new IDoc(idoc_pipeline);
```

Include the line

```
com.wm.pkg.sap.idoc.IDOC
```

under **Shared → Imports**.

For more information about using the IDoc Java API, see *Constructing an IDoc with the IDoc Java API* on page 9-4.

Converting an IDoc to XML

The following code sample shows how to convert an IDoc to XML.

```
idoc_pipeline.put("$encoding","corresponding character set of the IDoc");
Values out = Service.doInvoke("pub.sap.idoc", "encode", idoc_pipeline);
String xmlString = out.getString("xmlData");
```

Constructing an IDoc with the IDoc Java API

The following sample shows how to use the IDoc class to construct a new MATMAS IDoc. The first three lines are generated by Developer.

Remember to include com.wm.pkg.sap.idoc.* in *Shared* → *Imports*.

```
public static Values createIDOC(Values in)
{
Values out = in
  try {
      IDOC idoc = new IDOC();

       IDOCControl ctrl = idoc.getControlRecord();

       String docnum = "0000047112211178";
       String mandt = "000";
       ctrl.setField("DOCNUM", docnum);
       ctrl.setField("MANDT", mandt);
       ctrl.setField("DOCTYP", "MATMAS02");
       ctrl.setField("DIRECT", "1");
       ctrl.setField("RCVPOR", "BLABLA");
       ctrl.setField("RCVPRT", "LS");
       ctrl.setField("RCVPRN", "SAPADAPTER");
       ctrl.setField("SNDPOR", "SAPALR");
       ctrl.setField("SNDPRT", "LS");
       ctrl.setField("SNDPRN", "FCBCLNT000");
       ctrl.setField("CREDAT", "19990914");
       ctrl.setField("CRETIM", "153522");
       ctrl.setField("MESTYP", "MATMAS");
       ctrl.setField("IDOCTYP", "MATMAS02");
       ctrl.setField("SERIAL", "19990914153522");

       IDOCSegment seg1 = new IDOCSegment("E1MARAM");
       seg1.setSDATA("MSGFN", "005");
       seg1.setSDATA("MATNR", "BOXKEKSE");
       seg1.setSDATA("ERSDA", "19980318");
       seg1.setSDATA("ERNAM", "TOPSI");
       seg1.setSDATA("PSTAT", "KBG");
       seg1.setSDATA("MTART", "FERT");
       seg1.setSDATA("MBRSH", "L");
       seg1.setSDATA("MATKL", "G1113");
       seg1.setSDATA("MEINS", "PCE");
       seg1.setSDATA("BLANZ", "000");
       seg1.setSDATA("BRGEW", "0.550");
       seg1.setSDATA("NTGEW", "0.000");
       seg1.setSDATA("GEWEI", "KGM");
       seg1.setSDATA("VPSTA", "KBG");
```

```
 IDOCSegment seg2 = new IDOCSegment("E1MAKTM");
 seg2.setSDATA("MSGFN", "005");
 seg2.setSDATA("SPRAS", "D");
 seg2.setSDATA("MAKTX", "Schachtel mit Keksli");
 seg2.setSDATA("SPRAS_ISO", "DE");

 IDOCSegment seg3 = new IDOCSegment("E1MARCM");
 seg3.setSDATA("MSGFN", "005");
 seg3.setSDATA("WERKS", "0001");
 seg3.setSDATA("PSTAT", "BG");
 seg3.setSDATA("PLIFZ", "0");
 seg3.setSDATA("WEBAZ", "0");
 seg3.setSDATA("PERKZ", "M");
 seg3.setSDATA("AUSSS", "0.00");
 seg3.setSDATA("BESKZ", "E");
 seg3.setSDATA("AUTRU", "X");

 IDOCSegment seg4 = new IDOCSegment("E1MBEWM");
 seg4.setSDATA("MSGFN", "005");
 seg4.setSDATA("BWKEY", "0001");
 seg4.setSDATA("VPRSV", "S");
 seg4.setSDATA("VERPR", "0.00");
 seg4.setSDATA("STPRS", "15.50");
 seg4.setSDATA("PEINH", "1");
 seg4.setSDATA("BKLAS", "7920");
 seg4.setSDATA("VJVPR", "S");
 seg4.setSDATA("VJVER", "0.00");
 seg4.setSDATA("VJSTP", "15.50");
 seg4.setSDATA("LFGJA", "1998");
 seg4.setSDATA("LFMON", "04");
 seg4.setSDATA("PSTAT", "BG");
 seg4.setSDATA("KALN1", "000100126602");
 seg4.setSDATA("KALNR", "000100126603");
 seg4.setSDATA("EKALR", "X");
 seg4.setSDATA("VPLPR", "0.00");
 seg4.setSDATA("VJBKL", "7920");
 seg4.setSDATA("VJPEI", "1");
 seg4.setSDATA("BWPEI", "0");

 idoc.append(seg1); //appends the segment seg1 to the empty IDoc
 for (int i=0; i<idoc.size(); i++)
     System.out.println("Segment "+i+": "+idoc.getSegment(i));

idoc.insert("E1MARAM", seg3, true); //inserts seg3 after
                                    //"E1MARAM" as a child
 for (int i=0; i<idoc.size(); i++)
 System.out.println("Segment "+i+": "+idoc.getSegment(i));

idoc.insert("E1MARAM", seg2); //inserts seg2 after "E1MARAM"
                              //as the following segment is a
                              //child of "E1MARAM" it is
                              //inserted as a child, too
 for (int i=0; i<idoc.size(); i++)
     System.out.println("Segment "+i+": "+idoc.getSegment(i));

idoc.insert("E1MARCM", seg4); //inserts seg4 after "E1MARCM" on
                              //the same level
     for (int i=0; i<idoc.size(); i++)
          System.out.println("Segment "+i+": "+idoc.getSegment(i));

  } catch (Exception e) {
```

```
            System.out.println(e);
            e.printStackTrace();
        }
    }
```

Sending IDocs to a SAP System

The basic steps to send an outbound IDoc from either a client or a service are:

1. Create a transaction ID by invoking the pub.sap.client:createTID service.

2. Send the IDoc to the SAP server by invoking the pub.sap.client:invokeTransaction service, passing in your exports and tables.

   If an error occurs, repeat the pub.sap.client:invokeTransaction service invocation with the same transaction ID. The SAP System guarantees that the transaction will execute only once.

3. After pub.sap.client:invokeTransaction returns successfully, invoke pub.sap.client:confirmTID.

   The pub.sap.transport.ALE:OutboundProcess performs these steps. The following code sample shows how to use that to send an IDoc to an SAP system.

```
/* This sample code shows how to send an tRfc outbound call to
   an SAP system. It assumes, that you already have
   IDOC_DATA_REC_40/IDOC_DATA and
   IDOC_CONTROL_REC_40/IDOC_CONTROL in the pipeline. First the
   serverName (SAP alias) variable is put into the pipeline:
*/

// "in" pipeline contains the IDoc
    in.put("serverName", "B20");

    // Now send it to the SAP system called "B20"
    Values out =
Service.doInvoke("pub.sap.transport.ALE","OutboundProcess", in);
```

Transaction Features for HTTP and SAP-XML

When executing HTTP Posts using IDoc-XML and XRFC with transactions, you should make sure that your client correctly handles such calls. It should have transaction management that supports at least the following features:

- Transaction ID generation

- Resending documents with same transaction ID in error cases.

- After a successful execution of a transaction, a document cannot be sent again.

To transfer the transaction ID to the SAP adapter, you add an extra header to the HTTP POST, x-tid, that contains the transaction ID.

Example for 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>
```

Chapter 10: Security

Product Availability

There are two versions of the webMethods Integration Server:

- A version with weak encryption (40 bit).
- A version with strong encryption (128 bit) which will be discussed in this document. For productive use we strongly recommend to use this version of the Integration Server together with the SAP adapter.

Configuration

If you are already using the Integration Server with weak encryption, you can easily upgrade to the version with strong encryption by installing this version over the other one.

Specify settings before starting the listener.

Both the certificates and private keys are stored in the file system. Therefore, restrict access to these directories using operating system tools to ensure that the keys cannot fall into the wrong hands, or that non-trusted certificates can be stored.

- watt.iaik.serverSocketDebug=true
  debugs the HTTPS-Listener when receiving HTTPS-Requests
- watt.iaik.SocketDebug=true
  debugs HTTP-Client-Connections, when the Integration Server connects to an HTTPS-Server

webMethods Developer

You can also use SSL to secure the connection between the Developer and the Integration Server. This requires the following:

- The installed version of the Developer must support SSL.
- The Integration Server must be set up as an HTTPS server as described in the *webMethods Integration Server Administrator's Guide*.
- To open a connection to the server, the option "uses secure connection" and the corresponding SSL port in the Integration Server must be selected.
- According to the network architecture, you must determine whether a connection to the Integration Server is possible, and not separated by a firewall, for instance.
- User name and password must be entered for authentication. Authentication is also possible using a client certificate. For details, contact SAP.

Java Client Code

SSL can also be used when calling a service from Java client code. For this, the Developer must change the source code as follows:

A configuration file is not used on the client side and therefore all properties must be set in the static block:

```
   static {
// for trusted certificates
com.wm.util.Config.setProperty(„watt.security.CADir‟, „cadir‟ );
…
   }
```

A change in the context is required:

```
context.setSecure(true);
```

Initializing the Security Settings in the Integration Server

To initialize the security settings, see the *webMethods Certificate Toolkit Guide*, which can be found at *webmethods*\CertToolkit\doc\B2BCertToolkitGuide.pdf.

User Authentication Between SAP Adapter and an SAP System

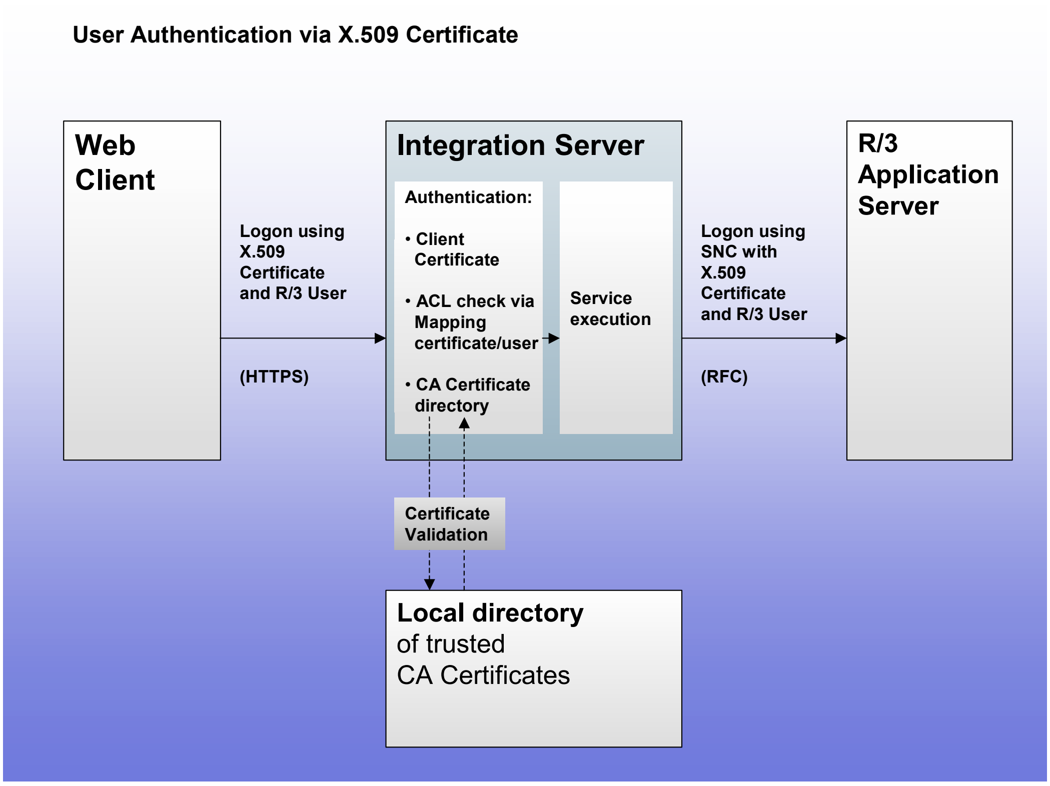
Authentication Through User Name and Password

When logging on through an HTTP or FTP client, standard user/password authentication is used, the user is mapped to a Integration Server session and eventually a service calling an SAP System will be executed. This service will use the logon parameters associated with an SAP System alias which will most likely not reflect the identity of the original HTTP client. The SAP user is much more a pool user shared among several physical users which allows optimal performance.

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.


User Authentication via X.509 Certificate
Web Client
Logon using X.509 Certificate and R/3 User
(HTTPS)
Integration Server
Authentication:
• Client Certificate
• ACL check via Mapping certificate/user
• CA Certificate directory
Service execution
Logon using SNC with X.509 Certificate and R/3 User
(RFC)
R/3 Application Server
Certificate Validation
Local directory of trusted CA Certificates

For the authentication via certificates against an SAP system it is required to enable SNC connections on your SAP adapter as well as on the SAP application server. These settings are listed in the section *'Defining SAP Servers (security options)* on page 4-3. For detailed information on the SAP application server and SNC client settings for SNC see the corresponding SAP documentation.

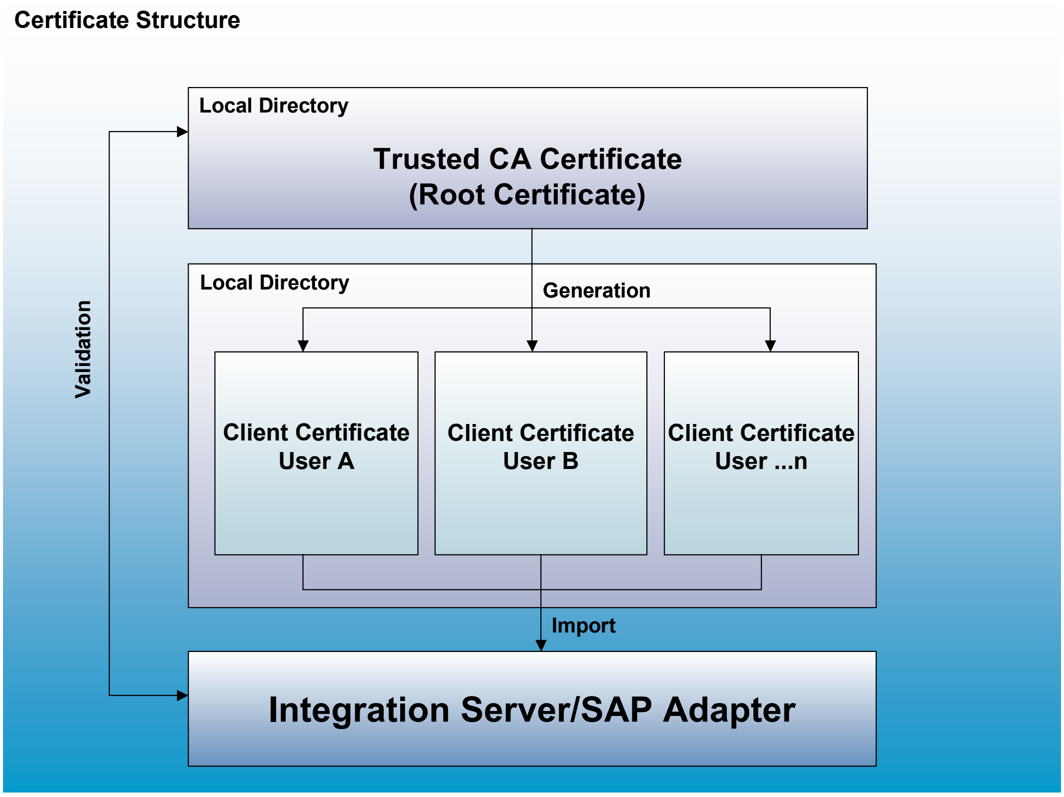
To define an HTTPS port that requests client certificates

1. Choose **Ports** from the **Security** section of the initial screen.
2. Click on the HTTPS port you want to edit.
3. Click **Edit HTTPS Port Configuration**.
4. Enter the option **Request Client Certificates** in the field **Client Authentication**.

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

If you want to log on to an SAP system via SAP adapter using any SAP user and a certificate, you can do so by providing a trusted certificate for the Integration Server. In a first step you have to request a signed certificate from a CA. For more information on how to request a certificate from a CA, see the *webMethods Certificate Toolkit Guide)*.

Having received the CA's certificate (root certificate), you can import the corresponding (personalized) client certificates for each user from a local directory to the Integration Server.



To import a client certificate from a local directory to the Integration Server

1. Choose **Certificates** from the **Security** section of the initial screen
2. Click on the link **Client Certificates**.
3. Enter the certificate path.
4. Click **Import Certificate**.

For validation purposes you also have to 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 installation directory.

To specify the CA certificate directory

5. Click on **Certificates** in the **Security** section of the initial screen.
6. Click on the link **Edit Certificate settings**.
7. Enter the path for the CA Certificate directory in the section **Trusted Certificates**.

If you leave the Trusted Certificates setting blank, the server will accept (trust) any certificate it receives during an outbound request (a certificate the server receives from a server that it submits a request to). However, the server *never* implicitly trusts a certificate for the purpose of authenticating an inbound request or validating an S/MIME signature. When you use either of these features, you *must* specify a Trusted Certificates directory, and that directory must contain the certificates of the CAs that your server trusts.

When a user logs on (e.g. from a Web client) using this certificate, the Integration Server verifies the root certificate in the CA Certificate directory and then passes the client certificate including the user name to the SAP system automatically. However, you have to make sure that this user can access the services he wants to execute in the Integration Server. That’s why you have to map the client certificate to the corresponding SAP adapter user or alternatively to a (standard) Integration Server user, depending on the authorizations required. If you want to execute a protected service within the Integration Server, the mapped user must be allowed in the corresponding ACLs (access control lists). For more information on ACLs see the *webMethods Integration Server Administration Guide*.

Example

You want to execute a service on the Integration Server that retrieves sales order data from an SAP system. This service is protected by an ACL. The Integration Server user ‘Sales’ is registered in this ACL and allowed to execute the service. If you map the certificate to the user ‘Sales’, your SAP (R/3) user can also execute the service. In addition, the SAP user must be authorized to execute the function modules of the corresponding function group.

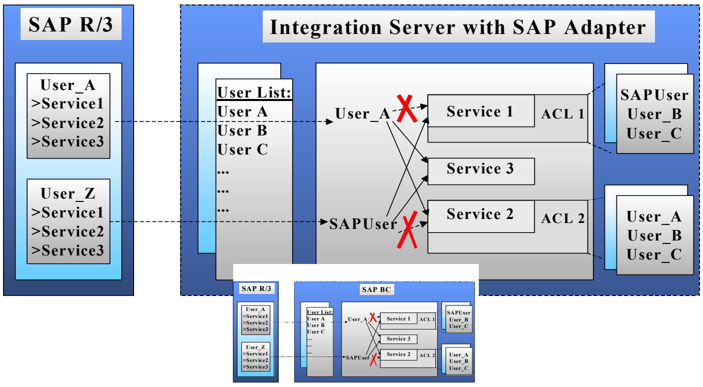
To map a client certificate to a Integration Server user

1. Choose **Certificates** from the **Security** section on the initial screen
2. Click on the link **Client certificates**.
3. In the section **Current Certificates** click on the certificate’s name you want to edit.
4. Click on the link **Change user mapping**.
5. Select a Integration Server user that is registered in the required ACLs.

To restrict the rights of the SAP logon users you should create specific user accounts in the SAP system with the minimum necessary set of authorizations. If for instance the Integration Server 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, see the SAP documentation.

Authentication When SAP Adapter acts as RFC-Server



If the SAP Adapter is called from an SAP system, the call is always trusted. Therefore, only the user name from the SAP System is used for the logon. This user is the user who triggered the synchronous or asynchronous call. If a user wants to execute a service protected by an ACL (access control list) this user must be entered in the corresponding ACL that allows access to this service.

In the new user concept for the SAP adapter 4.0 and higher the user **SAPUser** (Password: 22101999) is used as the default user instead of the user Default I.e., if the SAP adapter is called by an SAP user that does not exist within the Integration Server, the system switches automatically to the user SAPUser as default user.

The User SAPUser is part of the User Group SAPUsers and therefore also figures in the ACL *SAPUsers* that protects all Inbound Maps and Inbound Processes of the transports related to an SAP system against unauthorized access since 4.0.

The User *SAPUser*, as well as the User Group and the ACL are created automatically when you are starting the Integration Server server for the first time.

If an SAP user has been created in Integration Server in order to execute all Inbound Maps within the Integration Server, you have to assign this user at least to the User Group *SAPUsers*.

If you want to avoid that an SAP user which has not been created within the 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).

The user names in the Integration Server are case sensitive.

Authorization Configuration in the SAP Environment

For authorization configuration, see the webMethods Security Best Practices Guide.

Installing SAP Adapter According to Your Security Policy

The SAP adapter can only access SAP systems for which a SAP alias has been created in the *SAP server* list. There is no service available that allows you to execute RFC calls to SAP systems that are not defined there.

In addition to this restriction, you can also protect access to SAP systems in an intranet by installing an additional firewall between the Integration Server and the SAP Systems or by putting the Integration Server in the DMZ. You can configure the firewall to restrict which SAP Systems can be accessed from the SAP adapter through the SAP router.

Finally you might even want to completely disallow an Integration Server in the DMZ to actively open connections to an SAP system in the intranet. To do so, you need to install two Integration Servers, one in the DMZ and one in the intranet. The Integration Server in the DMZ can then be configured as a reverse invoke server. Thus the Integration Server in the intranet has to establish the connection to the reverse invoke server whereas the data still flows synchronously from the outside to the inside. For information on how to configure reverse invoke, see the *webMethods Integration Server Administrator's Guide.*

References

For further information on how to configure security, see the *webMethods Integration Server Administrator's Guide.*

Chapter 11: Managing the DDIC Cache

Data DICtionary Cache (DDIC Cache)

The Data DICtionary Cache (DDIC) is a cache that holds information about SAP function modules, structure definitions, Business Objects, and ALE mappings. The SAP adapter retrieves this information from an SAP server when it performs RFC or BAPI lookups. To improve performance, the SAP adapter caches information it receives about function modules and structure definitions.

The SAP adapter receives information about function modules and structure functions when:

- An RFC or BAPI lookup is requested from the Lookup screen
- A service that invokes an RFC or BAPI on an SAP server gets invoked on the Integration Server.
- An SAP server executes an RFC that invokes an Integration Server service

The SAP adapter keeps separate cached function modules, structure definitions, Business Objects, and ALE mappings for each SAP server. When the SAP adapter requires specific data, it checks its DDIC cache for the specific server to determine if the information it requires is in cache. If the required information is in the DDIC cache, the SAP adapter uses the cached information rather than retrieving it from the SAP system.

The DDIC cache is always active. There are no configuration tasks required to activate it.

When you are developing services, there may be times when the information in the DDIC cache becomes outdated, for example, if you change an RFC signature on an SAP server. In these situations, you can use the DDIC cache screens to view the information in the cache and remove specific function modules or structure definitions from the cache as necessary without having to restart the SAP adapter or the Integration Server.

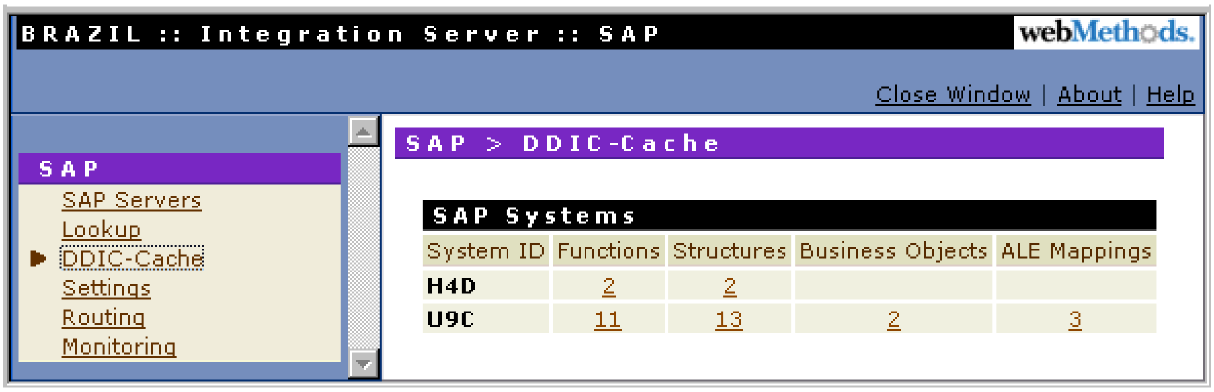
The DDIC cache does *not* persist through shutdown and restart of the SAP adapter or Integration Server.

Viewing Information in the DDIC Cache

Perform one of the following procedures to view information about the function modules, structure definitions, Business Objects, and ALE mappings that are in the DDIC cache.

To view information in the DDIC cache

1. Choose **SAP** from the Administrator UI navigation panel.
2. Select the **DDIC-Cache** tab if it is not displayed. The SAP adapter displays a screen that lists the number of function modules and structure definitions cached for each SAP server.



SAP Systems

System ID	Functions	Structures	Business Objects	ALE Mappings
H4D	2	2		
U9C	11	13	2	3

The System ID field is the name that was supplied when the SAP server was defined to the SAP adapter. If one or more SAP servers use the same system ID, they will share the same cache. For more information about defining SAP servers, see *Defining SAP Servers* on page 4-3.

3. To view the names of cached function modules for an SAP server, select the number in the **Functions** column for the appropriate SAP server.

4. To view the names of the cached structure definitions for an SAP server, select the number in the **Structures** column for the appropriate SAP server.

To view the function interface of cached function modules

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not already displayed.

3. To view the names of cached function modules for an SAP server, select the number in the **Functions** column for the appropriate SAP server.

4. Select the function module for which you want to see the function interface. The SAP adapter displays a screen that lists the function interface for the selected function module.

5. To view the structure definition of the displayed parameters, follow the hyperlink in the *Table* field. If you want to view the structures directly from the DDIC main page, follow the instructions below.

To view the structure definition of cached structures

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not displayed.

3. To view the names of the cached structure definitions for an SAP server, press the number in the **Structures** column for the appropriate SAP server.

4. Select the structure for which you want to see the structure definition. The SAP adapter displays a screen that lists the structure definition for the selected function module.

To view the cached business objects

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not displayed.

3. To view the names of the cached business objects for an SAP server, press the number in the **Business Objects** column for the appropriate SAP server.

To view the cached ALE mappings

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not displayed.

3. To view the names of the cached BAPIs for an SAP server (for which ALE information of an SAP System has been checked and extracted), press the number in the **ALE Mappings** column for the appropriate SAP server.

The list also contains BAPIs for which no ALE interface has been generated in the SAP system, but for which the availability has already been checked. This is because the SAP adapter tries only once to retrieve ALE information. If this retrieval fails, the SAP adapter remembers this fact.

Removing Information from the DDIC Cache

If a function module, structure definition, Business Object, or ALE mapping in the DDIC cache becomes outdated or corrupt, you can use the following procedures to remove it from the DDIC cache.

To remove function modules from the DDIC cache

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not displayed.

3. To view the names of cached function modules for an SAP server, select the number in the **Functions** column for the appropriate SAP server.

4. To remove a function module from the DDIC cache, press the ✕ icon in the **Remove** column.

To remove structure definitions from the DDIC cache

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not displayed.

3. To view the names of the cached structure definitions for an SAP server, press the number in the **Structures** column for the appropriate SAP server.

4. To remove a structure definition from the DDIC cache, press the ✕ icon in the **Remove** column.

To remove business objects from the DDIC cache

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not displayed.

3. To view the names of the cached business objects for an SAP server, press the number in the **Business Objects** column for the appropriate SAP server.

4. To remove a business object from the DDIC cache, press the ✖ icon in the **Remove** column.

To remove ALE mappings from the DDIC cache

1. Choose **SAP** from the Administrator UI navigation panel.

2. Select the **DDIC-Cache** tab if it is not displayed.

3. To view the names of the cached ALE mappings for an SAP server, press the number in the **ALE Mappings** column for the appropriate SAP server.

4. To remove an ALE mapping from the DDIC cache, press the ✖ icon in the **Remove** column.

Appendix A: Package Contents

Package Layout

All of the files relevant to the SAP adapter and its persistent state information are in the *Integration Server_directory*/packages/SAP directory. The following table describes the directories and files that make up the SAP package directory.

Directory	Contents
code	Contains the SAP adapter libraries.
config	Contains the following configuration files: sap.cnf. This file is created when you add your first SAP server using the SAP Administration interface. It stores server settings, listener settings, and maps. Back up this file before upgrading your server. cbr.cnf (content-based routing). This file contains the configuration parameter for for content-based routing (see p. 8-24).
logs	Contains the SAP adapter log files.
ns	Contains namespace information for the SAP adapter files.
pub, templates	Contains files and images comprising the SAP Administration interface.

Appendix B: Server Configuration

server.cnf

This file is created when you start the SAP adapter for the first time.

Some of the following statements can be added manually after installing SAP adapter.

Shut down the server and back up this file before you edit it.

The following statements are added automatically, when SAP adapter is started for the first time:

watt.sap.connection.poolSize	Maximal number of connections in one RFC connection pool to one SAP System (default: 10)
watt.sap.connection.timeout	Delay (minutes) until an unused connection to a SAP System is timed out (default: 5)
watt.sap.connection.timeoutCheckPeriod	Time interval in seconds between checks (seconds) whether unused pooled connections have timed out
watt.sap.connection.waitForPool	Delay (seconds) until requests waiting for a 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.
watt.sap.debug.level	Debug level from 1 to 10
watt.sap.debug.facList	List of Logging Facilities for the SAP Package.

Change/add the following statements to the server.cnf to increase performance:

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. (default=true) Set the value to "false" to remove all unnecessary whitespace and thus improve performance.

Change/add the following statements to the server.cnf to improve security:

watt.sap.snclibpath	Path of the SNC library needed for secure RFC connections.
watt.net.ssl.client.enforceTrust	When set to True, the server certificate is checked each time an SSL connection is established with the Integration Server as a client.

Change/add the following statement to the server.cnf to use a database as message store:

watt.PartnerMgr.xtn.store=db	Enable database logging

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

watt.sap.debug.dir	Set this variable to the directory in which you want to store the SAP log files (default: packages/SAP/logs)
watt.sap.throughput	Flag to decide whether to write basic throughput information to log. Valid values: true and false (default)
watt.sap.jarm/switch	Switch to turn jARM Monitoring 'on' (default) or 'off'

Appendix C: Representation of ABAP Types in the SAP Adapter

ABAP Type	Description	Dictionary Types	XML Representation
C	character[s]	CHAR, CLNT, CUKY, LANG, LCHR, UNIT	characters as string
D	Gregorian Calendar date	DATS	yyyy-MM-dd as string
F	IEEE double-precision 64-bit floating point	FLOAT	float as string
I	4-byte signed integer	INT4	integer as string
N	numeric character[s]	NUMC, ACCP	numeric characters as string
P	decimal number	CURR, DEC, QUAN	decimal as string
T	time	TIMS	HH:mm:ss as string
X	binary octets	RAW	base64 string
b	1-byte unsigned integer	INT1	integer as string
s	2-byte signed integer	INT2	integer in string
STRING	variable length string	STRING	string
XSTRING	variable length binary octets	RAWSTRING	byte[ ]

Appendix D: SAP Adapter API

SAP Client Services

pub.sap.client:connect

Establishes a connection to an SAP server.

Input Parameters

This key	Must specify…
serverName	SAP system alias to which the connection is established. The name must match a configured SAP server alias on the Integration Server.
$client (optional)	Client for the session. If no client is specified, the default client is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

This key	Contains…
result	"Ok" if connection was established successfully.

Example

Use the pub.sap.client:connect service when you want to establish a connection to a target SAP system. This might be useful for creating a session pool. In the most common scenarios, it is not necessary to invoke this service explicitly, as it is invoked implicitly (for example, when a pub.sap.client:invoke or pub.sap.client:invokeTransaction is issued).

pub.sap.client:lockSession

Locks an RFC connection to your Integration Server session so that you will always exclusively use this RFC connection for subsequent calls.

Input Parameters

This key	Must specify…
serverName	SAP system alias on which the session will be locked. The name must match a configured SAP server alias on the Integration Server.
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the user specified in the SAP system alias settings is used.
$pass (optional)	Password for the session. If no password is specified, the password specified in the SAP system alias settings is used.
$language (optional)	Language used during the session. If no language is specified, the language specified in the SAP system alias settings is used.

Return Values

There are no return values

Example

This service locks a session to trigger a *commit work* command inside the SAP System, which will cause a database commit and the start of the processing of posted data. This service only works with SAP Systems from 4.0A on, since BAPIs do not write data directly to the database but use the posting engine inside the SAP System. The data will not be written to the database until the client triggers a *COMMIT WORK* command. In order to call a BAPI which writes data, it is required to perform the following steps on the Integration Server:

1. Call the pub.sap.client:lockSession service in order to get an exclusive connection to the SAP System.

2. Perform the BAPI calls.

3. Call the pub.sap.bapi:commit or pub.sap.bapi:rollback service

4. Call the pub.sap.client:releaseSession service in order to release the exclusive connection to the SAP System and clean up used resources on the Integration Server

pub.sap.client:releaseSession

Releases a locked session on a SAP server.

Input Parameters

This key	Must specify…
serverName	SAP system alias on which the locked session is located. The name must match a configured SAP server alias on the Integration Server.
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

There are no return values

Example

The service will release a session after a commit work or rollback work command has been triggered inside the SAP System. This service only works with SAP Systems from 4.0A on, since BAPIs don't write data directly to the database but use the posting engine inside the SAP System. In order to call a BAPI which writes data, it is required to perform the following steps on the adapter:

1. Call the pub.sap.client:lockSession service in order to get a exclusive connection to the SAP System.

2. Perform the BAPI calls.

3. Call the pub.sap.bapi:commit or pub.sap.bapi:rollback service

4. Call the pub.sap.client:releaseSession service in order to release the exclusive connection to the SAP System and clean up used resources on the Integration Server

Example

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 server.

This service also needs the inputs (imports and tables) required by the function module.

Input Parameters

This key	Must specify…
serverName	Server alias from the server list on which the function module will be invoked.
$rfcname	Name of the function module to be invoked.
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

This service returns the outputs (exports and tables) returned by the function module.

This key	Contains…
$rfctime	Time (ms) spent within the RFC library to complete the invocation.
$runtime	Total invocation time (ms), including processing time within the adapter.

This key	Contains…
$encoding	MIME-compliant character set corresponding to the session's SAP code page.
$call	flag indicating whether pipeline represents a request (true) or a response (false) of a function module

Example

This service is used when you are directly calling an RFC on an SAP system without using an outbound map. (When an outbound map is used this service is invoked implicitly.)

pub.sap.client:invokeTransaction

Invokes an tRFC function module on a given SAP server..

Input Parameters

This key	Must specify…
serverName	SAP server alias for the SAP server on which you want to invoke the function module. The alias must match a configured SAP server alias on Integration Server.
$queueName (optional)	Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

This key	Must specify…
$runtime	Total invocation time (ms), including processing time within the adapter.
$rfctime	Time (ms) spent within the RFC library to complete the invocation.
$encoding	MIME-compliant character set corresponding to the session’s SAP code page.
$call	Flag indicating whether pipeline represents a request (true) or a response (false) of a function module.

Example

This service is used in outbound flows and can handle both synchronous and transactional RFC calls. Therefore, it can be seen as a generic wrapper for pub.sap.client:invoke, pub.sap.client:invokeTransaction and pub.sap.client:confirmTID.

pub.sap.client:createTID

Call this service if you want to obtain a transaction ID (TID; which is a GUID) that conforms to the format of SAP TIDs. The obtained TID can be used for pub.sap.client:invokeTransaction and pub.sap.client:confirmTID.

⚠ 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 the pub.sap.transport.ALE:OutboundProcess service rather than invoke pub.sap.client:createTID and pub.sap.client:invokeTransaction

Input Parameters

This key	Must specify…
serverName	SAP system alias on which the TID is created. This name must match a server list entry.
$queueName (optional)	Name of the SAP system inbound queue. Specify a value in case of a qRFC scenario
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

This key	Contains…
$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 an invokeTransaction in order to complete a transaction on the target system.

Input Parameters

This key	Must specify…
serverName	SAP system alias on which the transaction is confirmed. The name must match a server list entry.
$tid	Transaction ID to be confirmed.
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

There are no return values.

Example

This service is used in tRFC client scenarios in order to confirm a transaction that been executed completely.

- Invoke pub.sap.client:createTID. This service creates an SAP-conformant TID. It is necessary to have one, otherwise you cannot invoke the function module with pub.sap.client:invokeTransaction. If you have already obtained a TID, you should use that one in order to guarantee that the transaction is executed only once.

- Invoke pub.sap.client:invokeTransaction. This service invokes the given function module as a tRFC on the SAP system specified by serverName. The TID is passed as a parameter on this call.

- Invoke pub.sap.client:confirmTID. Confirms that the transaction has completed. Pass the same serverName and $tid value on this call.

pub.sap.client:getAttributes

Connects to the given SAP system and returns interesting information about this session.

Input Parameters

This key	Must specify…
serverName	SAP system alias on which the transaction is confirmed. The name must match a server list entry.
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

Record containing session information (RFC Attributes):

This key	Must specify…
ownHost	hostname (or IP) of the machine the adapter is running on
partnerHost	hostname (or IP) of the machine the SAP system is running on
systemNumber	system number of the SAP system
systemID	unique three-letter-ID of SAP system (in local network)
client	client the session is connected to
user	SAP user that has connected with this session
language	logon language
ISOLanguage	ISO name for logon language
ownCodepage	SAP codepage this connection is using
partnerCodepage	SAP codepage the SAP system is running on
ownRelease	version of the loaded RFC library used by the adapter

partnerRelease	release of the SAP system
kernelRelease	kernel release of the SAP system
partnerType	RFC type of the partner; R/3 (3), R/2(2) or external RFC server (E)
trace	flag indicating whether trace is turned on(true) or off(false) for this connection
ownType	RFC type of the adapter; should always be E
rfcRole	Role of the adapter in this call; should always be C (for client)
CPICConversationID	CPIC ID of the connection (low level protocol information)
encoding	IANA-encoding that is equivalent to the SAP codepage used, e.g. ISO-8859-1
charset	charset that is equivalent to the SAP codepage used, e.g. ISO8859_1 (used in Java constructors)
bytesPerChar	number of maximum bytes used per char in the codepage

pub.sap.client:getFunctionInterface

Looks up the function interface definition of an RFC function module in the given SAP system.

Input Parameters

This key	Must specify…
$rfcname	Function module for which the interface definition is looked up.
serverName	SAP system alias on which the transaction is confirmed. The name must match a server list entry.
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

Record list containing all parameter related information:

This key	Must specify…
paramClass	Class of parameter; I (importing), E(exporting), T(table), X(exception).
parameter	Name of the parameter in the function interface.
tabName	Name of the table in the SAP system the parameter is referring to.
type	Type of the parameter (e.g. CHAR, STRUCTURE, TABLE, ...)
length	Internal length in bytes of the parameter.
decimals	Only relevant for decimal types; number of decimals used.
optional	Flag indicating whether parameter is optional.

pub.sap.client:getStructureDefinition

Looks up the structure definition of an SAP structure in the given SAP system.

Input Parameters

This key	Must specify…
structName	Structure name for which the definition is looked up.
serverName	SAP system alias on which the transaction is confirmed. The name must match a server list entry.
$client (optional)	Client for the session. If no client is specified, the client specified in the SAP system alias settings is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

Record list containing all field related information.

This key	Must specify…
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 (e.g. CHAR, STRUCTURE, TABLE, ...).
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.

XRFC Services

pub.sap.rfc:encode

Converts an RFC call in a Values format to an XML string in a format specified by the SAP RFC-XML (XRFC) specification.

Input Parameters

This service needs the RFC calls in Values representation as input.

This key	Must specify…
repServerName or serverName	SAP system used as a repository. This name must match a server list entry.
$encoding	Specifies the encoding used for this strings, e.g. iso-8859-1
$call	This flag specifies whether the pipeline should be interpreted as a call (true) or as a response (false)
$envelope	bXML or RFC-XML, specifies the type of document generated.

Return Values

This key	Contains…
xmlData	Contains the RFC-XML representation of the pipeline as a string.
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, with HTTPAction). This is done in the XML transport's Outbound Process for function modules.

pub.sap.rfc:decode

Converts an RFC-XML (XRFC) string to a Values object that is in a format that can be passed to an SAP system.

Input Parameters

This key	Must specify…
bytes (optional)	Byte array that contains the data to be decoded as XRFC.
node (optional)	Document object that represents the XRFC data (you'll 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.
xmlData	A document in RFC-XML format as a string.
$encoding	Specifies the encoding used for this strings, e.g. iso-8859-1

Return Values

Values object corresponding to the RFC-XML document.

Example

This service can be used when an RFC-XML document is posted to the server in a variable.

pub.sap.rfc:createTemplate

Creates an RFC-XML (XRFC) template for the specified function module

Input Parameters

	Must specify…
repServerName	SAP server alias for the SAP server that is used as a repository. The alias must match a configured SAP server alias on the Integration Server.
$encoding	Specifies the encoding used for this strings, e.g. iso-8859-1
$rfcname	Function module for which you want to create the template.
$call	This flag specifies whether a call template (true) or a response template (false) is created.
$envelope	bXML or RFC-XML, specifies the type of template generated.

Return Values

This key	**Contains…**
xmlData	Contains the RFC-XML template as a string.

Example

Use this service to create templates for testing purposes. This service is included in the Lookup screen.

IDoc Services

pub.sap.idoc:encodeSDATA

Converts every SDATA field from orderly Values to byte array. This service is usually called prior to sending an IDoc to an SAP system via *pub.sap.client:invokeTransaction*.

Input Parameters

This service requires the following input parameters from the Values object.

Note that this service handles both IDoc versions 3 and 4. The difference between the two is that, for IDocs version 3, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 4, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

This key	Must specify…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is a Values object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field).
ServerName	Server alias which is used as repository for structure information about the IDoc.

Return Values

This key	Contains…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is a 1000-byte field. At this point, the IDoc in the pipeline is ready for sending to an SAP system via pub.sap.client:invokeTransaction.

Example

This service is most often used when sending an IDoc to an SAP system. Following is a sequence of service calls that take an IDoc in XML form, convert it to a Values object, then fire it into an SAP system:

- Invoke pub.sap.idoc:decode. This takes the input "xmlData" (the XML String) and creates a Values object that is in "RFC-ready form". (This means it matches the RFC call used to send an IDoc to an SAP system and that the pipeline is almost in the required format.)

- Invoke this service (pub.sap.idoc:encodeSDATA). This converts every SDATA field in each of the segments from orderly Values to a byte array. (The pipeline is ready for th final service that sends it to the SAP system.)

- Invoke pub.sap.client:invokeTransaction. This service invocation requires the pipeline (containing the IDoc Values object) and the serverName to send the IDoc to.

pub.sap.idoc:decodeSDATA

Converts the SDATA field from raw byte array to an orderly Values object (with keys and values corresponding to the segment table). This service is usually called immediately after receiving an IDoc from the SAP system in a Java service, for example.

Input Parameters

This service requires the following input parameters from the Values object.

Note that 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.

This key	Must specify…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. This is state of the Values object that is passed to a Java service from the SAP system. At this point, the SDATA field of each segment consists of a 1000-byte field.
serverName	Server alias which is used as repository for structure information about the IDoc.

Return Values

This key	Contains…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is now an orderly Values object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field).

Example

This service is most often used when receiving an IDoc from an SAP system. The following is a sequence of service calls that demonstrates how to convert an IDoc received in a Java service to the XML format specified by the SAP-XML specification:

- Invoke this service (pub.sap.idoc:decodeSDATA). This converts every SDATA field in each of the segments from a raw byte array to an orderly Values object. Now the IDoc and each of its segements has fully been converted to a Values pipeline object.

- 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:transformFlatToHierarchy

Reformats an IDoc from flat RFC format to hierarchical format which represents a hierarchical structure of an IDoc-XML document.

Input Parameters

This service requires the following input parameters from the Values object.

This service handles both IDoc versions 3 and 4. The difference between the two is that, for IDocs version 3, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 4, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

This key	Must specify…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC 40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is now an orderly Values object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field).
conformsTo (optional)	Name of the record structure of the IDoc. The record structure is used as a template to discriminate between records and record lists. Without *conformsTo*, all tables in the hierarchical structure will be record lists.

Return Values

This key	Contains…
boundNode	IDoc in hierarchical structure. Note that this is the format that the sap.idoc.transformHierarchyToFlat requires the IDoc as input.

pub.sap.idoc:transformHierarchyToFlat

Reformats an IDoc from hierarchical format to flat RFC format.

Input Parameters

This service requires the following input parameters from the Values object.

This key	Must specify…
boundNode	IDoc in hierarchical structure. Note, this is how the pub.web:documentToRecord service returns an IDoc.

Return Values

This service returns the IDoc in either version 3 or version 4 format. The difference between the two is that, for IDocs version 3, the service returns IDOC_CONTROL and IDOC_DATA. For IDocs version 4, it returns IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

This key	Contains…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC 40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is now an orderly Values object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field).

pub.sap.idoc.routing:registerService

This service registers a given service with the routing manager. The best way to use it is to invoke it in a replication or startup service.

Input Parameters

This key…	**Contains…**
msgType	The message type of the IDoc for which you want to register a content based routing or mapping. If the special value $default is used, a standard service will be registered.
service	The complete namespace reference of the service to register (format: folder.subfolder:servicename)
inbound	Flag that indicates whether to register an inbound (default) or an outbound service.

Return Values

This key…	**Contains…**
activated	Flag that indicates whether the service could be activated as routing service.
activeService	Only present if activated==false. In this case it contains the service that is currently active or $none if all registered services are deactivated.

pub.sap.idoc.routing:unregisterService

This service unregisters a given service from the routing manager.

Input Parameters

This key…	Contains…
msgType	The message type of the IDoc for which you want to unregister a content based routing or mapping. If the special value $default is used, a standard service will be unregistered.
service	The fully specified service to unregister
inbound	Flag that indicates whether to unregister an inbound (default) or an outbound service.

Return Values

This key…	Contains…
wasActive	Flag that indicates whether the unregistered service was the active routing service.

pub.sap.idoc:encodeString

Encodes an IDoc to a String equivalent, the format that is used by the file port of an SAP system.

When loading from a file, make sure that String is generated with correct encoding.

Input Parameters

This key…	Contains…
serverName	Server alias which is used as repository for structure information about the IDoc.
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is now an orderly Values object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field).

Return Values

This key…	Contains…
iDocString	The IDoc in the file port format as String.

pub.sap.idoc:decodeString

Encodes an IDoc to a String equivalent, the format that is used by the file port of an SAP system.

Input Parameters

This key…	Contains…
serverName	Server alias which is used as repository for structure information about the IDoc.
iDocString	The iDoc in the file port format as String.

Return Values

This key…	Contains…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is now an orderly Values object containing the keys and values from the segment table (the name of thesegment table is specified by the SEGNAM field).

IDoc-XML Services

pub.sap.idoc:encode

Service that converts an IDoc in a Values format to an XML string in a format specified by the SAP Idoc-XML Specification.

Input Parameters

This service requires the following input parameters from the Values object.

This service handles both IDoc versions 3 and 4. The difference between the two is that, for IDocs version 3, the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 4, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

This key	Must specify…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC 40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is now an orderly Values object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field).
$encoding	Specifies the encoding used for this strings; for example, ISO-8859-1.

Return Values

This key	Contains…
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. The following is a sequence of service calls that demonstrates how to convert an IDoc received in a Java service to the XML format specified by the SAP-XML specification:

- Invoke pub.sap.idoc:decodeSDATA. This converts every SDATA field in each of the segments from a raw byte array to an orderly Values object. Now the IDoc and each of its segments has fully been converted to a Values pipeline object.

- 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-XML Specification into an IDoc that is in a Values format necessary for an RFC call (using pub.sap.client:invokeTransaction).

Input Parameters

This service requires the following input parameters from the Values object.

This key	Contains…
bytes (optional)	byte array, that contains the data to be decoded as IDocXML.
node (optional)	Document object, that represents the IDocXML data (you'll 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.
xmlData	String. The IDoc in XML format. The XML format is consistent with the SAP-XML specification.
$encoding	Specifies the encoding used for this stings, e.g. iso-8859-1

Return Values

This service handles both IDoc versions 3 and 4.

If the tag for the IDoc control header is “EDI_DC” in the XML, this service converts it into a table called IDOC_CONTROL; this makes the IDoc a version 3 IDoc.

If the tag for the IDoc control header is “EDI_DC40” in the XML, this service converts it into a table called IDOC_CONTROL_REC_40; this makes the IDoc a version 4 IDoc.

This key	Contains…
IDOC_CONTROL IDOC_DATA or IDOC_CONTROL_REC40 IDOC_DATA_REC40	Both keys are Record list (Table) objects containing the control and data tables for the IDoc. The SDATA field is now an orderly Values object containing the keys and values from the segment table (the name of the segment table is specified by the SEGNAM field).

Example

This service is the first service called when you want to send an IDoc-XML document to an SAP system. The following ia a sequence of service calls that take an IDoc in XML form, convert it to a Values object, then fire it into the SAP system:

- Invoke this service (pub.sap.idoc:decode). This takes the input "xmlData" (the XML String) and creates a Values object that is in "RFC-ready form" (this means it matches the RFC call used to send an IDoc into the SAP system and that the pipeline almost in the required format).

- Invoke pub.sap.idoc:encodeSDATA . This converts very SDATA field in each of the segments from orderly Values to a byte array. Now the pipeline is ready for the final service that sends it to the SAP system.

- Invoke pub.sap.client:invokeTransaction. This service requires the pipeline (containing the IDoc Values object) and the *serverName* to send the IDoc to.

bXML Services

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

This key	Must specify…
$object	Name of the business object to encode.
$bapi	Name of the BAPI that is encoded.
$encoding(optional)	Specifies the encoding used for this strings, e.g. iso-8859-1.
$call(optional)	This flag specifies whether the pipeline should be interpreted as a call (true) or as a response (false)[default]
$abapexception(optional)	If this object is in the pipeline, an Exception document will be created.
$metabapi(optional)	Contains metadata information about the BAPI, that should be encoded.

Return Values

This key	Contains…
xmlData	A document in bXML format as a string.

pub.sap.bapi:decode

Decodes bXML documents and generates a corresponding pipeline.

Input Parameters

This key	Must specify…
xmlData (optional)	A document in bXML format as a string.
bytes (optional)	Byte array, that contains the data to be decoded as XRFC.
node (optional)	Document object, that represents the XRFC data (you'll get a node, e.g. when putting an XML file via FTP with extension .xml).
$encoding(optional)	Specifies the encoding used for the strings, e.g. iso-8859-1. The service checks in that order if an input document is available: xmlData, bytes, node.

Return Values

No return values.

pub.sap.bapi:createTemplate

Generates a bXML document template for the definition of a BAPI in an SAP system.

Input Parameters

This key	Must specify…
repServerName or serverName	SAP server alias for the SAP server that is used as a repository. The alias must match a configured SAP server alias on the Integration Server. Note: one of the parameters must be specified. As repServerName is primary, we recommend to use this parameter
$objectName	Name of the business object to encode.
$bapiName	Name of the BAPI that is encoded.
$encoding(optional)	Specifies the encoding used for this strings, e.g. iso-8859-1.
$call(optional)	This flag specifies whether a call template (true) or a response template (false) is created.

Return Values

This key...	Contains…
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.

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

This key	Must specify…
serverName	SAP system used. This name must match a server list entry.
$client (optional)	Client for the session. If no client is specified, the default client is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.
wait (optional)	Flag indicating if call should wait until data is really written to database.

Return Values

This key	**Contains…**
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.

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

This key	Must specify…
serverName	SAP system used. This name must match a server list entry.
$client (optional)	Client for the session. If no client is specified, the default client is used.
$user (optional)	User name for the session. If no user is specified, the default user is used.
$pass (optional)	Password for the session. If no password is specified, the default password is used.
$language (optional)	Language used during the session. If no language is specified, the default language is used.

Return Values

This key	Contains…
Return	Return code in BAPI style keeping the success/failure of the rollback.

Transport Services

pub.sap.transport.ALE:InboundProcess

Receives an IDoc and forwards it to the Partner Manager.

Input Parameters

This service requires the following input parameters from the Values object.

This service handles both IDocversions 3 and 4. The difference between the two is that, for IDocs version 3 the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 4, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

The SAP adapter parses the IDoc it receives from the SAP server and converts it into two Values List objects—one for the IDoc control and one for the IDoc body. The SAP adapter passes the Values List objects with the IDoc information to the Partner Manager to be routed

<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>$tid</td><td>Transaction ID (TID).</td></tr>
<tr><td>$action</td><td>One of the following transaction codes:
<table>
<tr><th>Code</th><th>Meaning</th></tr>
<tr><td>0</td><td>Create</td></tr>
<tr><td>1</td><td>Execute</td></tr>
<tr><td>2</td><td>Rollback</td></tr>
<tr><td>3</td><td>Commit</td></tr>
<tr><td>4</td><td>Confirm</td></tr>
</table></td></tr>
<tr><td>IDOC_CONTROL
IDOC_DATA

or

IDOC_CONTROL_REC_40
IDOC_DATA_REC_40</td><td>Record list (Table) object that contains the control and data tables for the IDoc.

The SDATA field is a Values object containing the keys and values form the segment table (the name of the segment table is specified by the SEGNAM field.)</td></tr>
</table>

Return Values

There are no return values.

pub.sap.transport.ALE:OutboundProcess

Sends an IDoc to an SAP server.

It is recommended that you use this service for IDocs rather than invoking pub.sap.client:createTID and pub.sap.client:invokeTransaction, which performs the same function. This service does not invoke pub.sap.client:confirmTID.

Input Parameters

This service requires the following input parameters from the Values object.

This service handles both IDoc versions 3 and 4. The difference between the two is that, for IDocs version 3 the service looks for IDOC_CONTROL and IDOC_DATA in the pipeline. For IDocs version 4, it looks for IDOC_CONTROL_REC_40 and IDOC_DATA_REC_40.

<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>transportParams</td><td>A record with the following key/value pair.</td></tr>
<tr><td>server</td><td>SAP server alias for the SAP server to which you want to send the IDoc. The alias must match a configured SAP server alias on the Integration Server.</td></tr>
<tr><td>IDOC_CONTROL
IDOC_DATA

or

IDOC_CONTROL_REC_40
IDOC_DATA_REC_40</td><td>Record list (Table) object that contains the control and data tables for the IDoc.

The SDATA field is a Values object containing the keys and values form the segment table (the name of the segment table is specified by the SEGNAM field.)</td></tr>
<tr><td>$tid</td><td>Transaction ID (TID).</td></tr>
<tr><td>$action</td><td>One of the following transaction codes:
<table>
<tr><th>Code</th><th>Meaning</th></tr>
<tr><td>0</td><td>Create</td></tr>
<tr><td>1</td><td>Execute</td></tr>
<tr><td>2</td><td>Rollback</td></tr>
<tr><td>3</td><td>Commit</td></tr>
<tr><td>4</td><td>Confirm</td></tr>
</table>
</td></tr>
</table>

Return Values

There are no return values.

wm.PartnerMgr.gateway.transport.B2B:InboundProcess

Receives a message from a remote Integration Server and forwards it to the Partner Manager. This service invokes the wm.PartnerMgr.gateway.runtime: InboundProcess service, which determines the routing rule to use to route the message based on the sender, receiver, and message type specified in the input parameters. After determining the routing rule to use, it invokes the flow service that processes the matching routing rule.

It is recommended that you use this service rather than directly invoking *wm.PartnerMgr.gateway.runtime:InboundProcess*, which performs the same function, since *wmPartnerMgr.gateway.transport.B2B:InboundProcess* stores the transaction into the Message Store..

Input Parameters

This service requires the following input parameters from the Values object.

In addition to the parameters specified below, the input parameters must also include the message to be routed through the Partner Manager. Use a key for the message that the outbound transport expects. For example, if the message is to be routed using the Service transport (wm.PartnerMgr.gateway.transport.B2B:OutboundProcess), use a key that the service that is to be invoked for this message recognizes.

This key	Contains…
sender	The sender of a message to be routed.
receiver	The receiver of the message.
msgType	Message type of the message.
$routeOnly	Optionally, whether the message is to be only routed or whether the message is to routed and recorded in the Message Store. Specify true if you want the message routed but not recorded in the Message Store. Specify false if you want the message both routed and recorded in the Message Store. If you do not specify this parameter, the default is false.
$tid	Optionally, the transaction ID. If you do not specify a transaction ID and $routeOnly is false, the Partner Manager generates a transaction ID for the transaction.
$action	Optionally, the transaction state. Specify one of the following codes:

Code	Gives the transaction this state:
0	CREATED
1	EXECUTED
2	COMMITTED
3	ROLLEDBACK

	4	CONFIRMED

Return Values

There are no return values.

wm.PartnerMgr.gateway.transport.B2B:OutboundProcess

Invokes a service on a local or remote Integration Server.

⚠ If you want to invoke a service on a remote Integration Server, the local Integration Server must have an alias defined for the remote server. For more information about created aliases for remote Integration Server, see the *webMethods integration Server Administrator's Guide*.

Input Parameters

This service requires the following input parameters from the Values object.

<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>$tid</td><td>Optionally, a transaction ID.

If you want to invoke the service using guaranteed deliver, specify the guaranteed delivery transaction ID. If you are not using guaranteed delivery, do not specify a transaction ID.</td></tr>
<tr><td>$action</td><td>Optionally, the transaction state. Specify one of the following codes:
<table>
<tr><th>Code</th><th>Gives the transaction this state…</th></tr>
<tr><td>0</td><td>CREATED</td></tr>
<tr><td>1</td><td>EXECUTED</td></tr>
<tr><td>2</td><td>COMMITTED</td></tr>
<tr><td>3</td><td>ROLLEDBACK</td></tr>
<tr><td>4</td><td>CONFIRMED</td></tr>
</table></td></tr>
<tr><td>transportParams</td><td>A record with the following key/value pair.
<table>
<tr><th>This key</th><th>Contains…</th></tr>
</table></td></tr>
</table>

serverAlias	Identification of the Integration Server on which the service is to be invoked. To invoke a service on a remote server, specify the alias for the remote server. The default value is local, which indicates the local server. Use local to invoke a service on the local server that is not password protected. To invoke a password protected service on the local server, set up an alias for the local server, specifying a user name and password that has access to the password protected service. Then, specify that alias name.
servicePath	Interface name for the service to invoke.
service	Name of the service to invoke.
valueScope	Optionally, the scope for the session. This is where you want the Partner Manager to store the connection to the remote server. 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.

Return Values

It returns the result of the service that is invoked.

wm.PartnerMgr.gateway.transport.EmailTransport:OutboundProcess

Sends a document via an e-mail message to specified recipients.

This service is intended for use only by the Partner Manager to route messages via e-mail messages. If you want to send information via e-mail outside of the Partner Manager, use the pub.client:smtp service.

This service is a wrapper for the pub.client:smtp service, which is in the WmPublic package.

Input Parameters

This service requires the following input parameters from the Values object.

<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>data</td><td>Values object that contains the outbound document to send in the e-mail message. The Values object contains the following fields:
<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>content</td><td>An object that represents the content of the document. It must be one of the following data types: String, byte[], InputStream.</td></tr>
</table></td></tr>
<tr><td>receiver</td><td>Receiver of the document.</td></tr>
<tr><td>msgType</td><td>Message type of the document.</td></tr>
<tr><td>transportParams</td><td>Values object containing the e-mail configuration parameters (all are of type String):
<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>contenttype</td><td>Content type of the document (e.g. MIME type).</td></tr>
<tr><td>to</td><td>E-mail address of the recipient of the document. To specify multiple recipients, separate each e-mail address with a comma.</td></tr>
<tr><td>cc</td><td>E-mail address of each recipient to receive a complimentary copy. To specify multiple recipients, separate each e-mail address with a comma.</td></tr>
<tr><td>bcc</td><td>E-mail address of each recipient to receive a blind complimentary copy. To specify multiple recipients, separate each e-mail address with a comma.</td></tr>
<tr><td>subject</td><td>Subject for the e-mail message.</td></tr>
<tr><td>from</td><td>E-mail address to use for the sender of the e-mail message.</td></tr>
</table></td></tr>
</table>

	mailhost	Host name of the SMTP server to which to send the e-mail message. By default, the SMTP server that the Integration Server uses for e-mail is used.

Return Values

This key	Contains…
status	A string indicating the status of the email outbound process, either “failed” or succeeded”.

wm.PartnerMgr.gateway.transport.FTPTransport:OutboundProcess

FTPs a document to specified host with the specified FTP host name, port number, user name, password, and directory path. The name of the document on the FTP server will have the format, ftpfilexx.data, where xx is an ID number that the FTP transport process defines. In case there is a tid in the pipeline, the filename is <tid>.xml

> This service is intended to be used only by the Partner Manager to route messages via FTP. If you want to send information via FTP outside of the Partner Manager, use the pub.client:ftp service.
>
> This service is a wrapper for the pub.client:ftp service, which is in the WmPublic package.

Input Parameters

This service requires the following input parameters from the Values object.

<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>data</td><td>A Values object that contains the outbound document to FTP. The Values object contains the following fields:
<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>content</td><td>An object that represents the content of the document. It must be one of the following data types: String, byte[], InputStream.</td></tr>
</table></td></tr>
<tr><td>receiver</td><td>Receiver of the document.</td></tr>
<tr><td>msgType</td><td>Message type of the document.</td></tr>
<tr><td>transportParams</td><td>Values object containing the FTP configuration parameters (all are of type String):
<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>serverhost</td><td>FTP host name where you want the document sent.</td></tr>
<tr><td>serverport</td><td>Port number on which the remote FTP server (serverhost) listens for incoming requests. The default number is 21.</td></tr>
<tr><td>username</td><td>Valid FTP user name that the Integration Server must supply to connect to the remote FTP server.</td></tr>
<tr><td>password</td><td>Corresponding password of the FTP user (username) that the Integration Server must supply to connect to the remote FTP server.</td></tr>
<tr><td>dirpath</td><td>Directory path on the target machine where the message is to be placed.</td></tr>
</table></td></tr>
</table>

Return Values

This key	Contains…
returncode	Return code from the FTP server. This is a three-digit number that indicates the status of the FTP put command.
returnmsg	Corresponding return message for the return code.
logmsg	Log message of the entire FTP session.

pub.sap.transport.RFC:InboundProcess

Receives an inbound RFC or an RFC-XML document that is to be routed through the Partner Manager. To route an RFC through the Partner Manager, the SAP System must add the SBCHEADER table to the RFC.

Input Parameters

This service requires one of following input parameters from the Values object.

<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>SBCHEADER</td><td>Array of records (key/value). For information about what to include in the array of records, see The SBCHEADER Table on page 5-12.</td></tr>
<tr><td>sbcHeader</td><td>Values object that represents the key/value pairs in the SBCHEADER table. For information about what to include in the array of records, see The SBCHEADER Table on page 5-12.</td></tr>
<tr><td>$tid</td><td>Optionally, a transaction ID.
If you want to process the RFC call asynchronously using tRFC, specify the transaction ID. If you would like to perform a synchronous RFC call, do not specify this parameter.</td></tr>
<tr><td>$action</td><td>Optionally, the transaction state. Specify one of the following codes:
<table>
<tr><th>Code</th><th>Gives the transaction this state:</th></tr>
<tr><td>0</td><td>CREATED</td></tr>
<tr><td>1</td><td>EXECUTED</td></tr>
<tr><td>2</td><td>COMMITTED</td></tr>
<tr><td>3</td><td>ROLLEDBACK</td></tr>
<tr><td>4</td><td>CONFIRMED</td></tr>
</table></td></tr>
</table>

Return Values

There are no return values.

pub.sap.transport.RFC:OutboundProcess

Invokes an RFC on an SAP server. This service can handle both RFC and tRFC calls.

Input Parameters

This service requires the following input parameters from the Values object.

<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>transportParams</td><td>A record with the following key/value pair.
<table>
<tr><th>This key</th><th>Contains…</th></tr>
<tr><td>server</td><td>The SAP server alias for the SAP server on which invoke the RFC. The alias must match a configured SAP server alias on the Integration Server.</td></tr>
</table></td></tr>
<tr><td>$tid</td><td>Transaction ID (optional).
If you want to process the RFC call asynchronously using tRFC, specify the transaction ID. If you would like to perform a synchronous RFC call, do not specify this parameter.</td></tr>
<tr><td>$action</td><td>Transaction state (optional). Specify one of the following codes:
<table>
<tr><th>Code</th><th>Gives the transaction this state:</th></tr>
<tr><td><b>0</b></td><td>CREATED</td></tr>
<tr><td><b>1</b></td><td>EXECUTED</td></tr>
<tr><td><b>2</b></td><td>COMMITTED</td></tr>
<tr><td><b>3</b></td><td>ROLLEDBACK</td></tr>
<tr><td><b>4</b></td><td>CONFIRMED</td></tr>
</table></td></tr>
<tr><td>$rfcname</td><td>(optional) RFC that should be called</td></tr>
</table>

Return Values

There are no specific return values, but the pipeline contains the result of the invoked function module.

pub.sap.transport.XML:InboundProcess

Can be called (for example, using HTTP post) with arbitrary XML documents to pass them on to the Partner Manager.

Input Parameters

This key	Contains…
node	Document object that can be generated by calling stringToDocument and is automatically generated when posting an XML document via HTTP (using text/xml).

Return Values

No specific values in the pipeline.

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 adapter or webMethods' Integration Server, you can use B2B transport**.**

This service requires the following input parameters from the Values object.

Input Parameters

This key	Contains…
xmldata(optional)	An XML document as string, only used if transportParams/xmlType is "Arbitrary XML".
transportParams	A record with the following key/value pair.

This key	Contains…
url	The URL to which to post the document.
xmlType	The XML dialect to use. Specify Values-XML if you want the XML in webMethods native XML format. Specify SAP-XML if you want the XML in a format that is compliant with the SAP XML specification. For an IDoc, IDoc-XML is used; for an RFC, RFC-XML (XRFC) is used. Note that the IDoc must be available as input to this service. New options: "Arbitrary XML" allows to post any XML document. ”SOAP XRFC” can be selected as additional XML dialect. This equals to XRFC (RFC-XML) with a SOAP envelope (higher than SOAP 1.1).
useTextXml (optional)	Flag that overwrites the content-type of bXML, XRFC and IDoc-XML to text/xml, if set to true.
useUTF8 (optional)	Flag that allows to force the XML rendering engines to use UTF-8 as encoding for the resulting documents.
httpUser	Optionally, the user name to supply for a user name/password authentification challenge.

	httpPassword	Optionally, the password to supply along with httpUser for a user name/password authentication challenge.
	useBapi	ON or OFF. If set to ON, the pipeline is encoded using bXML for Business Objects.
	object	The name of the business object, to which the call should be mapped.
	bapi	The name of the BAPI method, to which the call should be mapped
$tid	Optionally, the transaction ID. Specify a transaction ID if the document should be sent as a transaction.	

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 Partner Manager.

Input Parameters

This key	Must specify…
$object	Name of the business object that is represented in the pipeline.
$bapi	Name of the BAPI that is represented in the pipeline.
$sbcHeader	Record that contains the key/value pairs with routing information. Will be generated by the bXML parser.

Return Values

A pipeline that is representing the response or an exception of the executed BAPI.

pub.sap.transport.BAPI:OutboundProcess

Executes a BAPI in an SAP system. This transport service can handle both asynchnronous and synchronous execution of a BAPI.

Input Parameters

This key	Must specify…
$tid(optional)	see $tid for RFC:OutboundProcess.
$action(optional)	see $action for RFC:OutboundProcess.
transportParams	

This key	Must specify…
server	The SAP server alias for the SAP server on which execute the BAPI. The alias must match a configured SAP server alias on the Integration Server.
mode	Allows to restrict the way how the BAPI can be invoked: 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.

Return Values

A pipeline that represents the response or an exception of the executed BAPI.

Demo Services

sap.demo:handleIDocXMLPost

Submit an IDoc-XML document to ALE InboundProcess or Gateway InboundProcess on a adapter.

Input Parameters

This key	Contains…
xmlData (String)	XML string of the IDoc document to be sent. Required.

Return Values

None.

sap.demo:handleRfcXMLPost

Submit an RFC-XML to an SAP system to invoke the associated RFC function.

Input Parameters

This key	**Contains**…
repServerName (String)	the name for R/3 system. Required.
xmlData (String)	the XML string of the RFC-XML document. Required.
$envelope	bXML or RFC-XML, document type used for the post.

Return Values

This key	**Contains…**
xmlData	XML document representing the response of the function ....

➡ This service is designed only for executing function modules that do not require an explicit commit.

sap.demo:handlebXMLPost

Submits a bXML to an SAP system to invoke the associated BAPI.

Input Parameters

This key	Contains…
serverName	The SAP server alias for the SAP server on which execute the BAPI. The alias must match a configured SAP server alias on the SAP adapter.
xmlData (String)	XML document representing a BAPI request in bXML.
mode	Allows to choose the way how the BAPI will be invoked: sync: synchronous execution async: asynchronous execution routing: apply routing matching routing info in the document

Return Values

This key	Contains…
xmlData	XML document representing the response of the BAPI.

➡ This service is designed only for executing function modules that do not require an explicit commit.

sap.demo:transform_IDoc-RFC-Values_to_IDoc-XML

This is a flow service that invokes sap.idoc:view to retrieve and view an IDoc stored in the mailbox as XML, HTML, or in raw Values format.

Input Parameters

Same as those of sap.idoc:view.

Return Values

Same as those of sap.idoc:view.

sap.demo: 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 'sap/pub' directory. To call it create a routing rule, choose the B2B transport and specify this service or call it directly.

Input Parameters

None.

Return Values

None.

Sap.demo:invokeBAPIReturningbXML

This service is used within handlebXMLPost.

IDoc Java API

See *Integration Server_directory*\packages\SAP\doc\api\index.html.

Appendix E: List of Deprecated Services

In the following all services and specifications are listed that have been replaced by new ones with release 4.6. The old services are still available, but should be no longer used.

Previous	New service/specification
wm.PartnerMgr.gateway.transport.ALE:InboundProcess	pub.sap.transport.ALE:InboundProcess
wm.PartnerMgr.gateway.transport.ALE:OutboundProcess	pub.sap.transport.ALE:OutboundProcess
wm.PartnerMgr.gateway.transport.ALE:getRoutingInfo_Default	pub.sap.transport.ALE:getRoutingInfo_Default
wm.PartnerMgr.gateway.transport.RFC:InboundProcess	pub.sap.transport.RFC:InboundProcess
wm.PartnerMgr.gateway.transport.RFC:OutboundProcess	pub.sap.transport.RFC:OutboundProcess
wm.PartnerMgr.gateway.transport.BAPI:InboundProcess	pub.sap.transport.BAPI:InboundProcess
wm.PartnerMgr.gateway.transport.BAPI:OutboundProcess	pub.sap.transport.BAPI:OutboundProcess
wm.PartnerMgr.gateway.transport.XML:InboundProcess	pub.sap.transport.XML:InboundProcess
wm.PartnerMgr.gateway.transport.XML:OutboundProcess	pub.sap.transport.XML:OutboundProcess
wm.PartnerMgr.gateway.transport.XML:xmlRoutingInfo	pub.sap.transport.XML:xmlRoutingInfo
sap:confirmTID	pub.sap.client:confirmTID
sap:connect	pub.sap.client:connect
sap:createTID	pub.sap.client:createTID
sap.admin:getAttributes	pub.sap.client:getAttributes
sap.admin:getFunctionInterface	pub.sap.client:getFunctionInterface
sap.admin:getStructureDefinition	pub.sap.client:getStructureDefinition
sap:invoke	pub.sap.client:invoke
sap:indirectCall	pub.sap.client:invokeTransaction
sap:lockSession	pub.sap.client:lockSession
sap:releaseSession	pub.sap.client:releaseSession

Previous	New service/specification
sap.bapi:encode	pub.sap.bapi:encode
sap.bapi:decode	pub.sap.bapi:decode
sap.bapi.Browser:createTemplate	pub.sap.bapi:createTemplate
sap.bapi.transaction:rollback	pub.sap.bapi:rollback
sap.bapi.transaction:commit	pub.sap.bapi:commit
sap.rfc:encode	pub.sap.rfc:encode
sap.rfc:decode	pub.sap.rfc:decode
sap.rfc:createTemplate	pub.sap.rfc:createTemplate
sap.idoc:encode	pub.sap.idoc:encode
sap.idoc:decode	pub.sap.idoc:decode
sap.idoc:encodeSDATA	pub.sap.idoc:encodeSDATA
sap.idoc:decodeSDATA	pub.sap.idoc:decodeSDATA
sap.idoc:transformFlatToHierarchy	pub.sap.idoc:transformFlatToHierarchy
sap.idoc:transformHierarchyToFlat	pub.sap.idoc:transformHierarchyToFlat
sap.idoc.routing:inbound	pub.sap.idoc.routing:inbound
sap.idoc.routing:inboundDefault	pub.sap.idoc.routing:inboundDefault
sap.idoc.routing:outbound	pub.sap.idoc.routing:outbound
sap.idoc.routing:outboundDefault	pub.sap.idoc.routing:outboundDefault
sap.idoc.routing:registerService	pub.sap.idoc.routing:registerService
sap.idoc.routing:unregisterService	pub.sap.idoc.routing:unregisterService
sap:OutboundRFC	pub.sap.client:invoke *or* pub.sap.client:invokeTransaction *(for tRFC)*
sap:InboundRFC	*No replacment*

sap:disconnect	No longer necessary, because session pools are used
sap:tablesToRecordLists	No longer necessary, because tables can be handled in flow now natively

Index