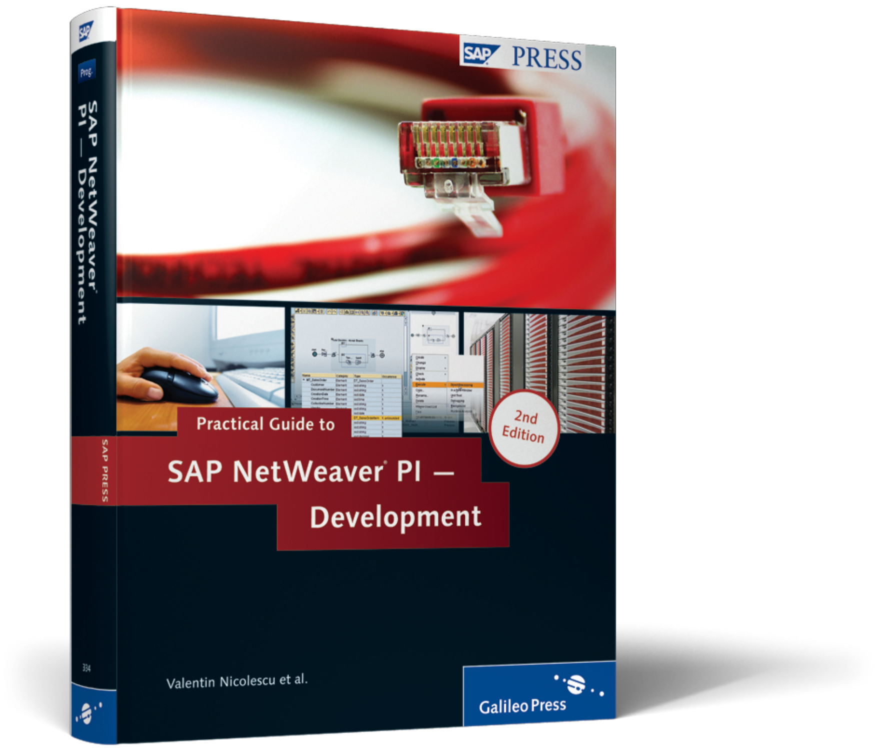
Valentin Nicolescu, Burkhardt Funk, Peter Niemeyer, Matthias Heiler, Holger Wittges, Thomas Morandell, Florian Visintin, Benedikt Kleine Stegemann, and Harald Kienegger

Practical Guide to SAP NetWeaver® PI – Development



www.sap-press.com

Galileo Press

Bonn • Boston

Contents at a Glance

Contents

PART II Basic System Configuration

The exercises in this chapter show you how to use SAP NetWeaver Process Integration (PI) components by presenting scenarios that are linked in content, but technically independent, to prepare you for the case study in the following chapter.

4 Technical Exercises

Using the available concepts and adapters of SAP NetWeaver PI is the basis for implementing complex integration scenarios. This chapter shows you how to configure adapters, create mappings, and monitor scenarios. The individual exercises build on each other and get more complex to help you gain the knowledge necessary to implement the case study presented in Chapter 5, SARDIS Case Study in Sales and Distribution.

Although the individual exercises depend on each other, you can use the lists of every exercise to track which objects are reused so you can start with a more advanced lesson. Predefined objects aren't used in the exercises, so you can reproduce all of the steps for completing the integration scenario at any time.

Appropriate for several participants

All of the exercises are designed in such a way that they can be performed by members of a class at the same time. Still, some steps can only be carried out once. Any steps that must be performed by the instructor either prior to or during the course will be noted.

Note

Although the exercises are appropriate for workgroups, you can also complete them on your own. If you do, you should also perform the steps that would be implemented by the instructor.

Even if you are going to complete the exercises alone, we recommend using a user number, as this simplifies comparisons between your work and the described procedure. In this case, you should use the instructor's number — 00.

Develop your own applications

Most exercises are completed as a development consultant, and, particularly in the beginning of the exercise block, you will assume the role of the system administrator or lead developer. In some places, you will have the opportunity to develop your own small applications in ABAP or Java. You hardly need any prerequisites for this, because the sample listings can be found in Appendix A, Exercise Materials, and in digital form on the website for this book (*http://www.sap-press.com*).

Exercises to prepare for the case study

The exercises deal with selected adapters and aspects of the PI environment. Various elements played a role in selecting the integration scenarios. On one hand, we present adapters that enable the presentation of reproducible exercises. On the other, we identify aspects that are necessary to implement the case study.

Even though the individual exercises do not have to be completed in the given order, we did choose their order for a reason. In this chapter, you will implement integration scenarios that will prepare you for the case study in Chapter 5. Using PI messages, you will create material in another system and verify its success. In the final step, the creation of material master data is reported to the person responsible for all materials. A business process ensures that these reports are delivered in bundles per agent.

Exercise procedures

To begin, you will use an ABAP program in system A, which records the data of a material to be created and transfers this data to the PI system using the remote function call (RFC) adapter (see Section 4.1, Exercise 1: RFC-to-File). There, the material master record is converted to a file in PI format via the file adapter. In the next exercise, this file is read by the file adapter and converted into an intermediate document (IDoc), which is transferred to and directly processed on system B (see Section 4.2, Exercise 2: File-to-IDoc).

In the third example, you will check to ensure that the material has been created successfully. Based on an ABAP proxy, you will send a call to the PI system. This request is converted to a Web service call provided by system B. The response of this control is synchronously returned to calling system A (see Section 4.3, Exercise 3: ABAP-Proxy-to-SOAP). The agent uses an ABAP program to report the successful creation of the material master records using a business process (see Section 4.4, Exercise 4: Business Process Management). As an alternative to the design of the second exercise, the example from the second scenario will be readdressed in a fifth scenario, replacing the receiving IDoc adapter through a Java database connectivity (JDBC) database (see Section 4.5, Exercise 5: File-to-JDBC).

Even though the contents of the individual exercises are based on one another, you can start with any of the exercises by using the appropriate templates.

4.1 Exercise 1: RFC-to-File

Course of the first exercise

In the first exercise, you will use your own ABAP program to call a remote-enabled function module that transfers material master data via the RFC adapter to the PI system. Once there, the data is converted to PI XML format and stored as a file. To keep things simple, the file is created directly on the PI server's file system.

Although the file technically remains on the PI system, from a logical viewpoint you will configure the receiving file adapter for system B. The communication in this integration scenario is asynchronous, because no business response is returned after sending the material data. The roles of systems A and B, and the adapters used in this exercise, are illustrated in Figure 4.1.

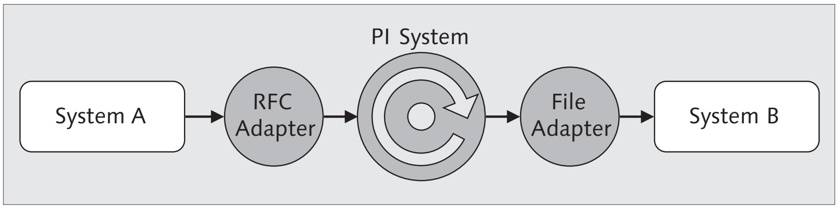


Figure 4.1 Scheme of Exercise 1: RFC-to-File

4.1.1 Basic Principles

New ABAP components

Because this book does not focus on the development of ABAP programs and remote-enabled function modules, we will only give you a basic explanation of the program and function module used. You can get an appropriate transport with the function module and the program for 20 participants and 1 instructor from the book's web page (*http://www.sap-press.com)* and implement it in your system A. For implementing the transport, consult your landscape administrator, if necessary.

If you want to create the program and the function module yourself, you will find the corresponding sample source code in Appendix A.

Structure of the function module

First, log on to the client of system A as the user SYS_A-##. There, you can view the remote-enabled function module using the Function Builder in Transaction SE37. Select the function module Z_RFM_MATERIALINPUT_##, where ## is your participant number.

You will see that from the function module's point of view the parameters are only imported and no value is returned. This is one of the two prerequisites for asynchronous communication from a sending RFC adapter.

Except for the interface definition, the function module does not contain any ABAP code. This means that the function module is used as a kind of dummy that forwards the transferred data to SAP NetWeaver PI and serves as an interface. The information about where to forward the data transferred to the function module is contained in the calling program.

Function of the ABAP program

The program Z_PROG_MATERIALINPUT_##, which you can find with the function module in the same transport request, and that you can view in Transaction SE38, has two functions: First, it accepts the basic material master data that will be used to create a new material in system B. Second, it calls the function module (described earlier) with the parameters listed in Table 4.1. The naming of these parameters is explained in the second exercise, in Section 4.2.

In this call, two things need to be mentioned: The remote-enabled function module is called to a specific destination (i.e., in the system behind this RFC connection). In the case of destination SystemA_Sender-##, this is the PI system, so the values transferred to the function module are forwarded to the PI system. The second aspect is the call in the background that makes the communication asynchronous.

Transferred Data	Description
MATNR	Material number
MAKTX	Material description
ERSDA	Creation Date (will be added automatically)
ERNAM	User name of creator (will be added automatically)
MTART	Material type
MBRSH	Industry sector
MATKL	Material group
MEINS	Quantity unit
BRGEW	Gross weight
GEWEI	Weight unit
MTPOS_MARA	General item category group

Table 4.1 Data Transferred to the Function Module Z_RFM_MATERIALINPUT_##

4.1.2 Design

Creating the design objects in the Enterprise Services Repository

At first, you need to create the various data and message types, and the service interfaces, with the required mappings in the Enterprise Services Repository. In a later phase of the configuration, these elements will be linked to the connected business systems (system A and system B).

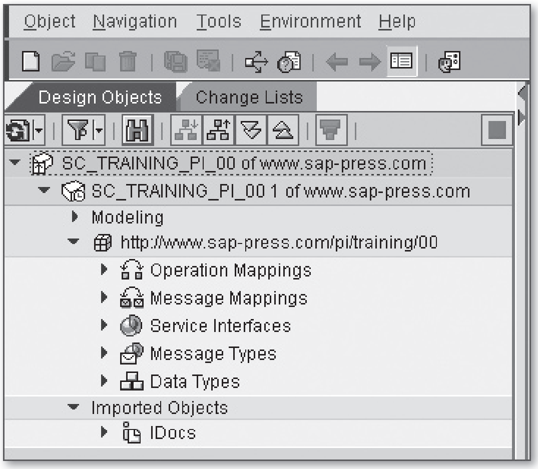
First steps

First, call Transaction SXMB_IFR from one of the connected systems or from the PI system itself. This opens the PI tools menu in your web browser, which should look familiar to you if you prepared for the exercises (see Chapter 3, Basic System Configuration). At the top-left, select the entry to the Enterprise Services Repository.

After the Java Web Start application has been updated and you have logged into the PI system as the appropriate user, the user interface of the Enterprise Services Repository is displayed. Make sure that you do not log on using the initial password; instead, change it during the logon to the SAP GUI.

On the left side, you will find the software components that have already been declared. This includes the software component `SC_Training_PI_##` with the namespace `http://www.sap-press.com/pi/training/##`, which is where you will store your elements in the Enterprise Services Repository.

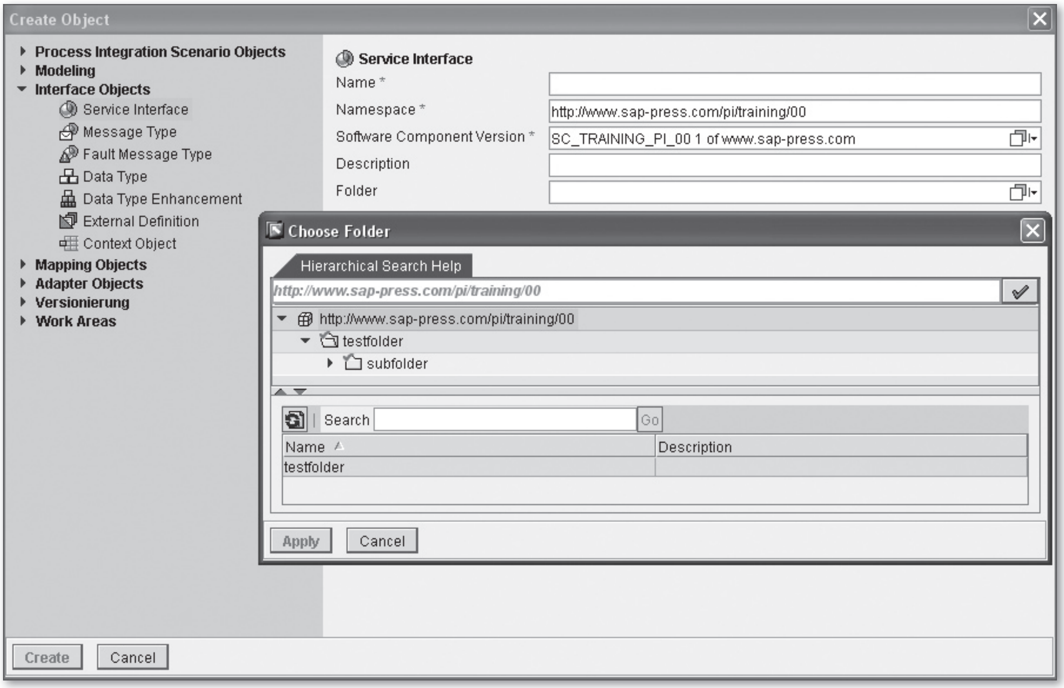
For a better overview, restrict the view to your software component. In the tree structure, click on your software component, and, above the tree, click the ONLY DISPLAY SELECTED SUBTREE icon. The Enterprise Services Repository should then look like Figure 4.2.



SC_TRAINING_PI_00 of www.sap-press.com
SC_TRAINING_PI_00 1 of www.sap-press.com
Modeling
http://www.sap-press.com/pi/training/00
Operation Mappings
Message Mappings
Service Interfaces
Message Types
Data Types
Imported Objects
IDocs

Figure 4.2 Entry to the Enterprise Services Repository

Folders have been introduced with SAP NetWeaver PI 7.1. Their use is optional, and serves the organizational division of design and configuration elements. Folders can be used in both the Enterprise Services Repository and the Integration Builder; the Enterprise Services Repository folder is created via the context menu of a namespace. It is possible to assign design elements directly to a folder when they are created. Folders are logically associated with a namespace. Figure 4.3 shows the namespace `http://www.sap-press.com/pi/training/00` in the selection window and the folder TESTFOLDER directly underneath. This folder is in turn associated with the sub SUBFOLDER. In the selection window, you can choose either the superordinate or the subordinate folder.



Service Interface
Name *
Namespace * http://www.sap-press.com/pi/training/00
Software Component Version * SC_TRAINING_PI_00 1 of www.sap-press.com
Description
Folder

http://www.sap-press.com/pi/training/00
- http://www.sap-press.com/pi/training/00
  - testfolder
    - subfolder

Name	Description
testfolder	

Figure 4.3 Creation of Folders in the Enterprise Services Repository

If you decide to detail the development structure beyond the presented structure, you can assign already-existing elements to a folder via drag and drop. For each of the folders, you can create a substructure in the form of subfolders. These are not limited by their number.

Another advantage of this concept is the authorization administration. You can assign authorizations to the folders on the group, role, and user levels. However, because authorization management is not the focus of this book, we won't discuss it here in detail.

An overview of the elements required for this exercise is given in Table 4.2. The roles of individual elements and their connections have already been explained in Chapter 2, SAP NetWeaver PI.

Object Type	Sender Side	Receiver Side
Service interface	Z_RFM_ MATERIALINPUT_##	SI_Material_Asnyc_ In
Message type		MT_Material
Data type		DT_Material
Operation mapping	OM_Z_RFM_MATERIALINPUT_##_to_SI_Material_ Async_Out	
Message mapping	MM_Z_RFM_MATERIALINPUT_##_to_MT_Material	

Table 4.2 Elements in the Enterprise Services Repository for the First Exercise

Solutions to interrupted connections

Note

If the connection to the Enterprise Services Repository is interrupted while an object is being edited, you can click on the Administration option in the Process Infrastructure (PI) tools and release the locked object for editing in the Lock Overview area.

Creating Elements for the Sending System

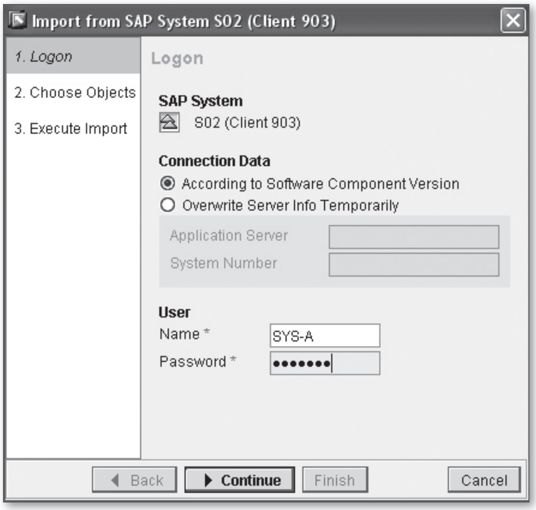
Design objects on the sender side

By using an RFC adapter, this scenario has a particular aspect: all elements on the sender side are replaced with the interface definition of the RFC module. The interface is imported from system A and not created in the Enterprise Services Repository to accelerate work and reduce the error rate.

To import the RFC interface, expand the bottom directory, Imported Objects, right-click to open the context menu, and find the Import of SAP Objects function. In the following window, select the According to Software Component Version option in the Connection Data area, because the system data has already been stored (see Figure 4.4). If this option is not available, enter the host name and the system number (system A). Next, enter your user SYS_A-## and the appropriate password before continuing.

Import the RFC interface

The next step lets you choose between RFC and IDoc interfaces. Expand the RFC option, and all remote-enabled function modules in system A are determined and displayed. Because this data collection can take a while, when you perform these steps in a group you can import all of the interfaces before starting the exercise. From the list, select function module Z_RFM_MATERIALINPUT_## (see Figure 4.5) and continue with the import.

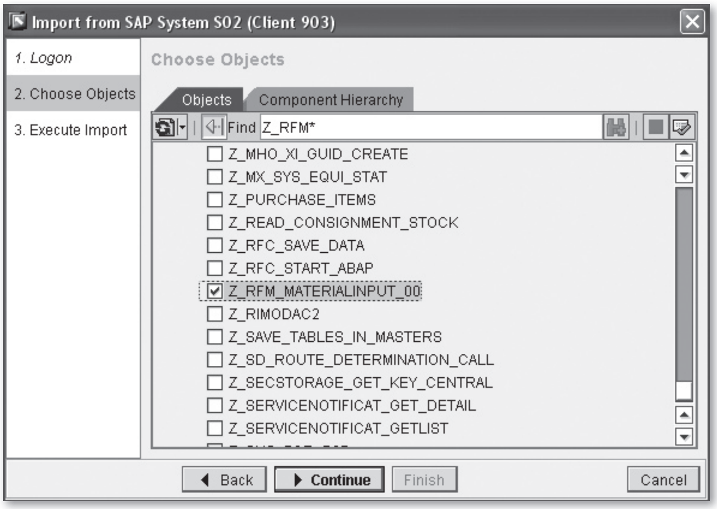


SAP System
S02 (Client 903)

Connection Data
[✓ According to Software Component Version]
Overwrite Server Info Temporarily
Application Server
System Number

User
Name * SYS-A
Password * •••••••

Figure 4.4 Import of RFC Interfaces — Login



Find Z_RFM*

- ☐ Z_MHO_XI_GUID_CREATE
- ☐ Z_MX_SYS_EQUI_STAT
- ☐ Z_PURCHASE_ITEMS
- ☐ Z_READ_CONSIGNMENT_STOCK
- ☐ Z_RFC_SAVE_DATA
- ☐ Z_RFC_START_ABAP
- ☑ Z_RFM_MATERIALINPUT_00
- ☐ Z_RIMODAC2
- ☐ Z_SAVE_TABLES_IN_MASTERS
- ☐ Z_SD_ROUTE_DETERMINATION_CALL
- ☐ Z_SECSTORAGE_GET_KEY_CENTRAL
- ☐ Z_SERVICENOTIFICAT_GET_DETAIL
- ☐ Z_SERVICENOTIFICAT_GETLIST

Figure 4.5 Import of RFC Interfaces — Selection

In the final step of the import process, check your selection and finish the import. After the import has completed, you can see your newly imported interface for your software component version in the IMPORTED OBJECTS • RFC directory. It is marked with a separate icon that indicates that this element has not been activated yet.

Creating Elements for the Receiving System

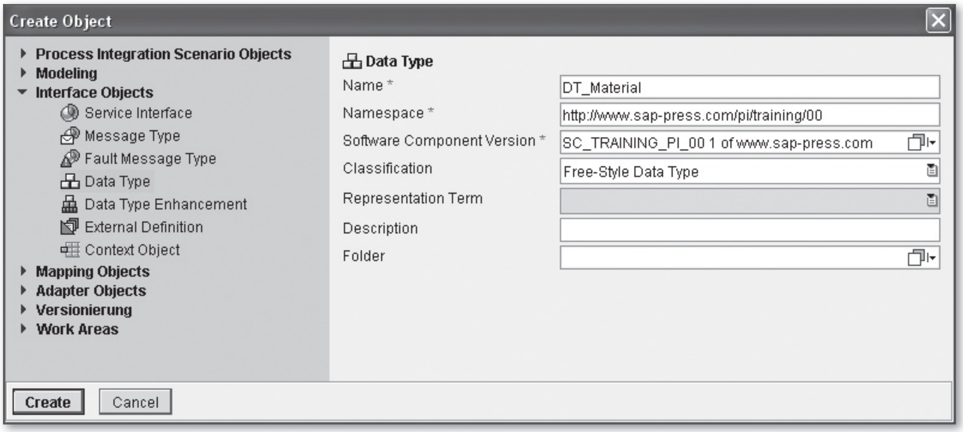
Design objects on the receiver side

While all of the elements are created on the side of the sending system by importing the RFC interface, you will create a data type, a message type, and a service interface for the receiving system. We recommend beginning with the independent elements (i.e., with those on the lowest hierarchy level), which, in this case, is the data type.

Creating a data type

Within your namespace, expand the DATA TYPE directory and open the creation dialog via the New entry of the context menu. In this window, you can enter the name of the new object, along with a description (see Figure 4.6).

The namespace and the software component version are automatically completed because you called the dialog in the appropriate context. Also, it is important to note the left area of this screen, which lists the elements that can be created within the Enterprise Services Repository. You can change what kind of element you want to create at any time. You will see a similar structure later when working in the Integration Directory.



Data Type

Name *	DT_Material
Namespace *	http://www.sap-press.com/pi/training/00
Software Component Version *	SC_TRAINING_PI_00 1 of www.sap-press.com
Classification	Free-Style Data Type
Representation Term	
Description	
Folder	

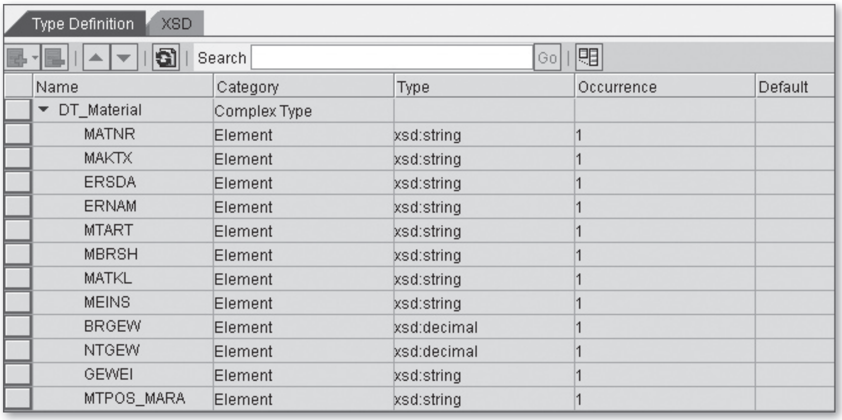
Figure 4.6 Dialog for Creating an Object in the Enterprise Services Repository

Name the new data type `DT_Material` and click on CREATE. The details window of a new data type is displayed on the right side. Because the structure of this window is typical of all detail views in the Integration Builder, it is used to explain some functions.

Next to the menu line of the Enterprise Services Repository is a separate details menu; the most important functions of which are also displayed as icons to its right. In addition to the icons for switching between the display and changing mode, and for creating a copy, you will also find an icon for the where-used list of this element (for example). The icon group to the right allows you to control the view; for example, you can hide header data or detach the details window as an independent element.

Structure of the DT_Material data type

In the case of your data type, the lower area of the details window contains a list of all data type elements. Using the relevant icons, you can add new rows to the top of the table and enter the elements from Figure 4.7. Please note that this is the type `xsd:string`. Only the `BRGEW` element has the type `xsd:decimal`; you will perform a calculation using this value later. In addition, add the missing element `NTGEW` of the type `xsd:decimal`. You will use this element to calculate the net weight of the material (based on the gross weight) in the message mapping. Save the data type after all of the elements have been inserted.



Name	Category	Type	Occurrence	Default
DT_Material	Complex Type			
MATNR	Element	xsd:string	1	
MAKTX	Element	xsd:string	1	
ERSDA	Element	xsd:string	1	
ERNAM	Element	xsd:string	1	
MTART	Element	xsd:string	1	
MBRSH	Element	xsd:string	1	
MATKL	Element	xsd:string	1	
MEINS	Element	xsd:string	1	
BRGEW	Element	xsd:decimal	1	
NTGEW	Element	xsd:decimal	1	
GEWEI	Element	xsd:string	1	
MTPOS_MARA	Element	xsd:string	1	

Figure 4.7 Editing a Data Type

Creating a message type

Because data types in the PI environment are used exclusively for modularizing data formats, and cannot appear in a mapping or interface themselves, they are embedded in message types. While data types can only be assigned to message types in a 1:1 ratio, data types can be combined in any ratio.

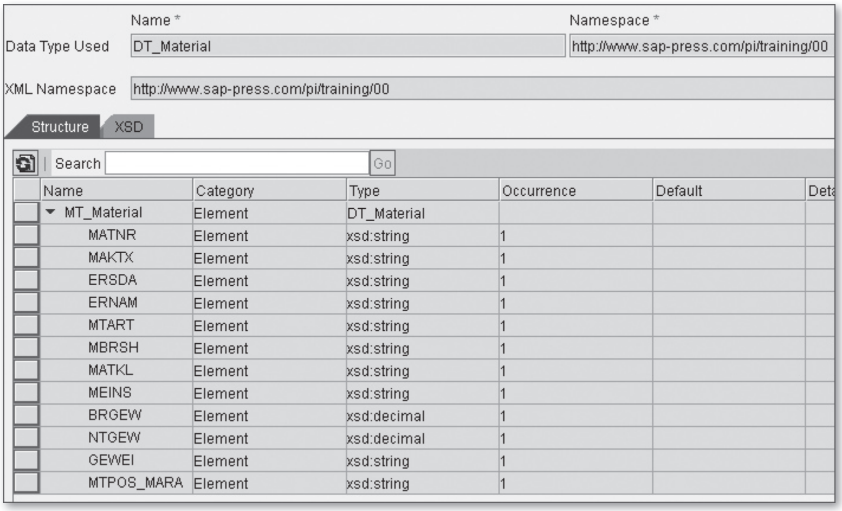
To create a message type, open the appropriate context menu by right-clicking on the Message Types directory. Select the New option. The familiar creation dialog box is displayed, this time for a message type. Name the new object `MT_Material` and enter a description. Continue with the detail view by clicking Create. Pay attention to the Data Type Used area in the middle; this is where you should insert the data type you just created. You have three ways of doing so:

Methods for selecting the data type

- The most obvious method is typing the name and the namespace; however, this involves the risk of typos.
- The second option is to select the object in an ABAP-based SAP system, such as in the input help. To do this, click on the hand and question mark icon to the right of the namespace field. A window opens, containing all of the data types created in your software component version for selection. The NAME and NAMESPACE fields are then populated.
- The third option is to drag and drop the selection. This is particularly suitable if your software component version contains a lot of data types, but there are only a few in your namespace. You can also pick the data type from the directory structure to the left and drop it on the hand next to the namespace field. Only by dropping it over the hand can you ensure a correct data transfer.

As you can see, all three ways work, even without activating the data type.

After selecting the appropriate data type, the lower area of the details window shows the structure of the used data type (see Figure 4.8). Check the structure and save the message type.



Data Type Used — Name *: DT_Material — Namespace *: http://www.sap-press.com/pi/training/00

XML Namespace: http://www.sap-press.com/pi/training/00

Structure | XSD

Search [ ] Go

Name	Category	Type	Occurrence	Default	Deta
▾ MT_Material	Element	DT_Material			
MATNR	Element	xsd:string	1		
MAKTX	Element	xsd:string	1		
ERSDA	Element	xsd:string	1		
ERNAM	Element	xsd:string	1		
MTART	Element	xsd:string	1		
MBRSH	Element	xsd:string	1		
MATKL	Element	xsd:string	1		
MEINS	Element	xsd:string	1		
BRGEW	Element	xsd:decimal	1		
NTGEW	Element	xsd:decimal	1		
GEWEI	Element	xsd:string	1		
MTPOS_MARA	Element	xsd:string	1		

Figure 4.8 Editing a Message Type

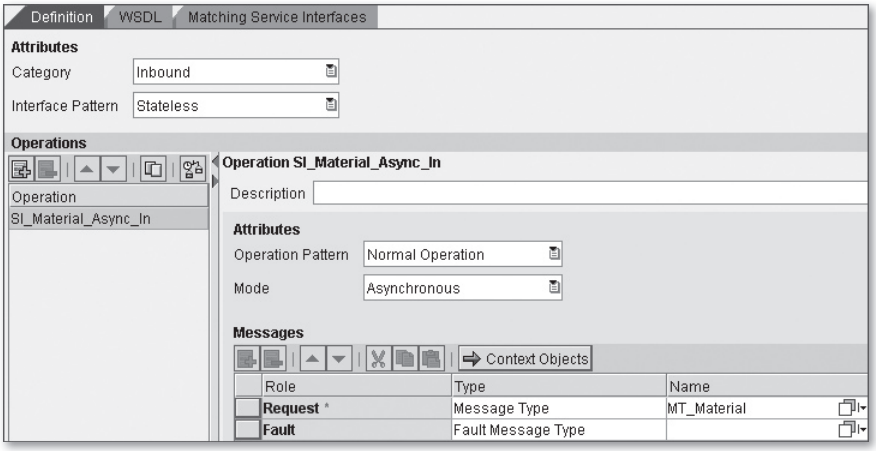
Creating a service interface

The last object on the receiver side is the service interface, which determines if a message can be received or sent, and whether the message is sent synchronously or asynchronously.

To create this service interface, open the context menu of the corresponding directory. Enter the name `SI_Material_Async_In` and an appropriate description, and then click Create to get to the details window. You can choose from the Inbound, Outbound, and Abstract options for the interface category; the individual categories were discussed in Chapter 2.

Because we are dealing with the interface on the receiver side, select INBOUND. The communication mode determines whether a response regarding the contents is expected or not. Because this is a one-way scenario, select the ASYNCHRONOUS mode.

You probably noticed that the input options for message types change every time the attributes are modified. You should now see the fields for the REQUEST MESSAGE TYPE and the FAULT MESSAGE TYPE. However, you will only use the former (see Figure 4.9). Using one of the three methods discussed earlier, select the message type `MT_Material` as the input message, and then save your service interface.



Attributes

Category	Inbound
Interface Pattern	Stateless

Operations

Operation
SI_Material_Async_In

Operation SI_Material_Async_In

Description

Attributes

Operation Pattern	Normal Operation
Mode	Asynchronous

Messages

Role	Type	Name
Request *	Message Type	MT_Material
Fault	Fault Message Type	

Figure 4.9 Editing a Service Interface

Creating the Mapping Objects

The connection between the elements of the sending and the receiving side is established via mapping. The contents conversion of the data formats in the form of message mapping is embedded in the operation mapping that connects a pair of inbound and outbound interfaces.

Creating the message mapping

To begin, create the message mapping. In your namespace, open the context menu of the Message Mappings directory. In the creation dialog, enter the name `MM_Z_RFM_MATERIALINPUT_##_to_MT_Material`, where ## represents your participant number. Choose a description and create the object.

Selecting the outbound and the target message

The center area of the details window is divided into two parts, allowing you to select the sending message type on the left side and the receiving message type on the right side. First, start with the message type on the sending side: You can either use the input help, or drag the appropriate message type to the Enter a source message label. In this exercise, there is no explicit message type on the sender side, so use the RFC interface.

Regardless of the selection method, you must choose which RFC message you would like to use. This is because synchronous communication is expected for an RFC interface. Therefore, you can choose between

`Z_RFM_MATERIALINPUT_##` and `Z_RFM_MATERIALINPUT_##.Response`. Select the former because no response is expected.

The left part of the center area now lists the elements of the RFC interface. For the receiving part, select your `MT_Material` message type.

Methods for mapping elements

If you look at the elements on the right side, you'll find a red mark next to every entry. This indicates an incomplete mapping for the respective target element. Because we didn't change anything in the OCCURRENCE column when creating a data type, the default value of **1..1** is applied. This means that this element is mandatory. If one of the target fields does not receive a value from the mapping, an error occurs. The connection between the elements of the two message types can also be established via three different methods:

- The most obvious method is connecting via drag and drop, where it isn't important which side is dragged to the other. The two elements are displayed in the lower screen area and connected automatically.
- The second option is to double-click on the source and target element to move them to the lower screen area, where they are displayed as rectangles. There you can connect the two rectangles by dragging the white area of the sending element to the corresponding area of the receiving element. This method should be used if the mapping is extended by predefined functions.
- The third method is suitable for connecting a large number of elements of the same name. To do this, parent elements must be selected on both sides. In this mapping, these are `Z_RFM_MATERIALINPUT_##` on the sender side, and `MT_Material` on the receiver side. Then, above the sending message type, select the MAP SELECTED FIELDS AND SUBSTRUCTURES IF NAMES ARE IDENTICAL icon. An alert dialog box appears, asking you to confirm the action. After dismissing the dialog, all of the elements on the sender side are connected to those on the receiver side. It's important to note that mapping of element names is case sensitive.

Graphical function for calculating the net weight

Perform a mapping using the third method, and have the result displayed in the overview by clicking on the DEPENDENCIES icon. The two message types then move apart and give way to the display of connection lines.

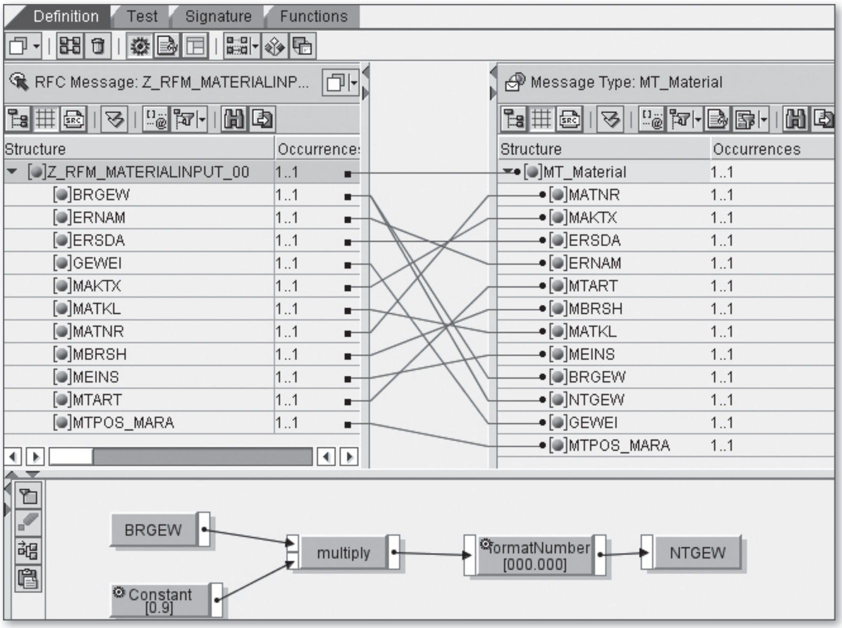
You will also notice that the marks next to the receiver elements have now turned green. Only the `NTGEW` element is still red, because it was not automatically provided with a value.

For demonstrating the integrated mapping functions, we assume that the net weight of the material is 90% of the gross weight. To map this, first select the `NTGEW` element on the receiver side, and then the `BRGEW` element on the sender side, by double-clicking on these items so both are displayed in the bottom area. To make this calculation, you first need a multiplication function that calculates the net weight from the gross weight with a constant of 0.9.

In the toolbar at the bottom of the screen, select the functions for Constants and click on the Constant function to the right, which is then displayed as a rectangle in the work area of the mapping. The cog wheel means that you can maintain parameters within this function. Double-click on the constant rectangle and change its value to 0.9 for the 90% of the net weight.

Now change to the Arithmetic area in the toolbar to insert the multiply function in the work area. Connect the `BRGEW` and `Constant 0.9` elements to the MULTIPLY function by dragging the white subareas. In fact, these functions would be sufficient for calculating the correct net weight. However, the three decimal places permitted for the `xsd:decimal` type might be exceeded. If this message mapping were tested, it would result in an error.

Before the result of the calculation can be mapped to the `NTGEW` element, it must be formatted using the FORMATNUMBER function from the Arithmetic functional area. Configure the internal parameter NUMBER FORMAT of the function so that the result matches the scheme `000.000`. Insert the FORMATNUMBER function between the MULTIPLY function and the target element `NTGEW` (see Figure 4.10). All rectangles and the mark next to the `NTGEW` target element should now be colored green. Save the message mapping.



Structure	Occurrence
Z_RFM_MATERIALINPUT_00	1..1
BRGEW	1..1
ERNAM	1..1
ERSDA	1..1
GEWEI	1..1
MAKTX	1..1
MATKL	1..1
MATNR	1..1
MBRSH	1..1
MEINS	1..1
MTART	1..1
MTPOS_MARA	1..1

Structure	Occurrences
MT_Material	1..1
MATNR	1..1
MAKTX	1..1
ERSDA	1..1
ERNAM	1..1
MTART	1..1
MBRSH	1..1
MATKL	1..1
MEINS	1..1
BRGEW	1..1
NTGEW	1..1
GEWEI	1..1
MTPOS_MARA	1..1

Figure 4.10 Message Mapping of the RFC-to-File Exercise

Testing the mapping

To ensure that the new mapping works, a test function is added to the Enterprise Services Repository, which you can select via the TEST tab in the top area of the details window. The left side of the test area displays the structure of the sending message type whose elements are populated with test values. Be sure to use a decimal point as the decimal character for the `BRGEW` element.

The test itself is started with the START THE TRANSFORMATION icon (indicated by a vise) at the bottom-left of the test area. If the test program does not find any errors, the structure of the receiving message type with its respective values is displayed on the right. In particular, you should verify whether the `NTGEW` element has been populated correctly.

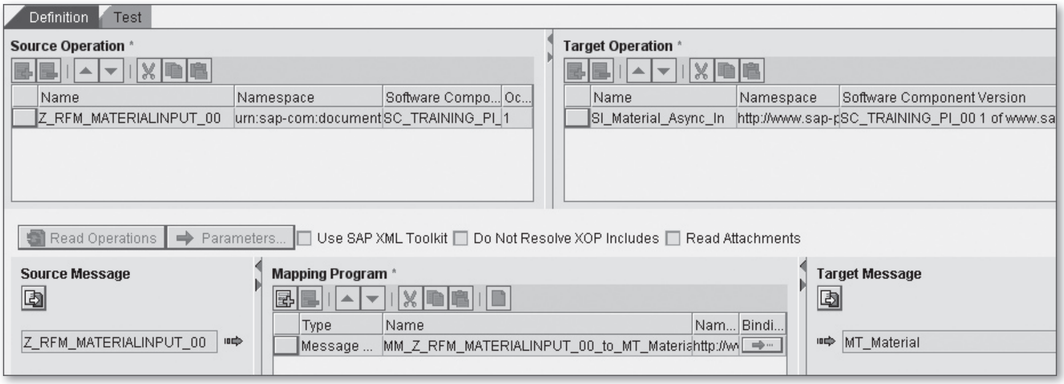
Creating the operation mapping

The operation mapping is the last object of the integration scenario you are creating in the Enterprise Services Repository. Start the creation dialog by opening the context menu of the Operation Mappings directory in your namespace. Name the operation mapping `OM_Z_RFM_MATERIALINPUT_##_to_SI_Material_Async_In`, and then enter a description for the object. Create it by clicking the Create button.

This object's detailed view is divided into an upper interface area and a lower mapping area. In the upper interface area, select the sender interface; this is the RFC interface `Z_RFM_MATERIALINPUT_##`. Note that the RFC interface is not stored in your namespace; instead, you will find it via the IMPORTED OBJECTS • RFC menu path. Perform the same steps for the `SI_Material_Async_In` target interface.

By selecting the two interfaces, you have now specified which interfaces will communicate with each other and which message types are used. However, you still need to determine how the two data formats are converted to each other, because there might be different message mappings for the same message pair.

In the lower mapping area, click the READ OPERATIONS button to display the message types of the used interfaces (see Figure 4.11). After the SOURCE and TARGET OPERATION fields have been filled, click on the NAME field located between the source and target message fields and select the input help that appears. A list is displayed, which contains all message mappings that exist between the interfaces in this sender and receiver scenario; you should only see mappings of the scheme `MM_Z_RFM_MATERIALINPUT_##_to_MT_Material`. Select the mapping with your member number.



Source Operation *

Name	Namespace	Software Compo...	Oc...
Z_RFM_MATERIALINPUT_00	urn:sap-com:document	SC_TRAINING_PI_	1

Target Operation *

Name	Namespace	Software Component Version
SI_Material_Async_In	http://www.sap-p	SC_TRAINING_PI_00 1 of www.sa

Source Message: Z_RFM_MATERIALINPUT_00

Mapping Program *

Type	Name	Nam...	Bindi...
Message ...	MM_Z_RFM_MATERIALINPUT_00_to_MT_Materia	http://w	

Target Message: MT_Material

Figure 4.11 Operation Mapping of the RFC-to-File Exercise

If you take a closer look at the MAPPING PROGRAM area, you will notice that the tabular structure allows you to select several mappings. All selected message mappings are processed sequentially according to their order in the table. When creating the message mapping, for example, you can use the TEST tab to perform a test which, in addition to the message mapping, also checks interface compatibility. Save your operation mapping after it has been successfully tested.

Activating the new design objects

As you can see, all newly created objects were usable throughout the entire Enterprise Services Repository, even though they were not activated. However, you can't access all of these objects in the Integration Directory in this state, so the next step is to activate your change.

To do this, go to the directory structure on the left and select the Change Lists tab. The tree structure is hidden, and your software component version is displayed, which you should fully expand. Beneath that list, you will find a Standard Change List containing all newly created objects. Verify that all elements presented in Table 4.2 are included in the change list.

Select the Activate option from the change list's context menu. A window containing all of the objects of the list is displayed. You have the option of excluding specific objects from the activation, but activate the entire list and return to the Objects tab. Notice that the icons indicating that the new objects are not yet activated have disappeared.

Note

You can also activate individual items via their context menu.

4.1.3 Configuration

First steps in the Integration Directory

Based on the objects created in the Enterprise Services Repository, you can now set up communication between systems A and B in the Integration Directory. The Integration Directory can be called by Transaction SXMB_IFR, by a direct link in the web browser, or by following the ENVIRONMENT • INTEGRATION BUILDER menu path in the Enterprise Services Repository.

As with the Enterprise Services Repository, the interface is divided into two parts; however, the objects are no longer arranged according to software component versions. Instead, they are arranged according to object types. Above the directory structure, you'll see three tabs: Change Lists, Objects, and Scenarios. The Change Lists tab serves the same function as in the Repository. The Objects tab lists all objects of the Directory by their type. Except for the scenario, which you will create for all of your objects in the Integration Directory; this exercise uses all of the elements listed in Table 4.3.

Object Type	Sender Side: System A	Receiver Side: System B
Communication channel 1	`RFC_ Senderchannel_##`	`File_ Receiverchannel_##`
Sender agreement	`| SystemA | Z_RFM_ MATERIALINPUT_## | |`	
Receiver agreement		`| SystemA | | SystemB | MI_ Material_Async_In`
Receiver determination	`| SystemA | Z_RFM_MATERIALINPUT_##`	
Interface determination	`| SystemA | Z_RFM_MATERIALINPUT_## | | SystemB`	

Table 4.3 Elements in the Integration Directory for the RFC-to-File Exercise

Setting Up the Business Systems and Their Communication Channels

Creating a configuration scenario

To create the `PI_Training_##` scenario, use the context menu of an existing scenario, or click the Create Object icon on the bottom left of the menu bar. Save the object so the scenario is displayed in the listing on the left side, select the new scenario, and then restrict the view by clicking on the Only Display Selected Subtree icon above the list. Creating a configuration scenario serves the organizational division of configuration objects.

Follow the Communication Component • Business System menu path. Below the branch, you will see at least two business systems, SystemA and SystemB, which were declared in the System Landscape Directory (SLD) during the preparations for the exercises. Click the Assign Configuration Scenarios option in the context menu of system A, and select the scenario you just created.

Due to this mapping, this business system and its communication channels are displayed in your scenario. Repeat this step for business system B. You need to configure a sending RFC adapter for system A and a receiving file adapter for system B.

Creating an RFC sender channel

Using the context menu from the Communication Channel path, open the creation dialog and enter system A as the Communication Component, the name `RFC_Senderchannel_##`, and an appropriate description. In the details window, use the input help to set the adapter type to RFC. Select the SENDER direction for this adapter. In the TRANSPORT PROTOCOL field, choose the RFC entry. For the MESSAGE PROTOCOL and ADAPTER ENGINE fields in the upper area, just use the default values.

Connection to the existing RFC connection

The RFC SERVER PARAMETER area establishes a TCP/IP connection to the RFC destination on the side of system A. During the preparation of the exercises (in Chapter 3, Section 3.5.2, Settings for the Use of the RFC Adapter), you created an RFC connection named `SystemA_Sender-##`. This RFC connection is registered on the gateway server of the PI system and is waiting for a corresponding counterpart.

In the APPLICATION SERVER field, enter the host name of the PI system, and in the APPLICATION SERVER SERVICE field, enter the gateway service of the PI system according to the scheme `sapgwXX`, where *XX* represents the instance number. The PROGRAM ID follows the scheme `SystemA_Sender-##` and, like the two values mentioned earlier, exactly matches the values entered in the corresponding RFC connection in system A. The SNC option specifies whether communication over an RFC connection takes place via a secure network connection (SNC). The UNICODE checkbox must be enabled if system A is a Unicode system.

Access to RFC metadata

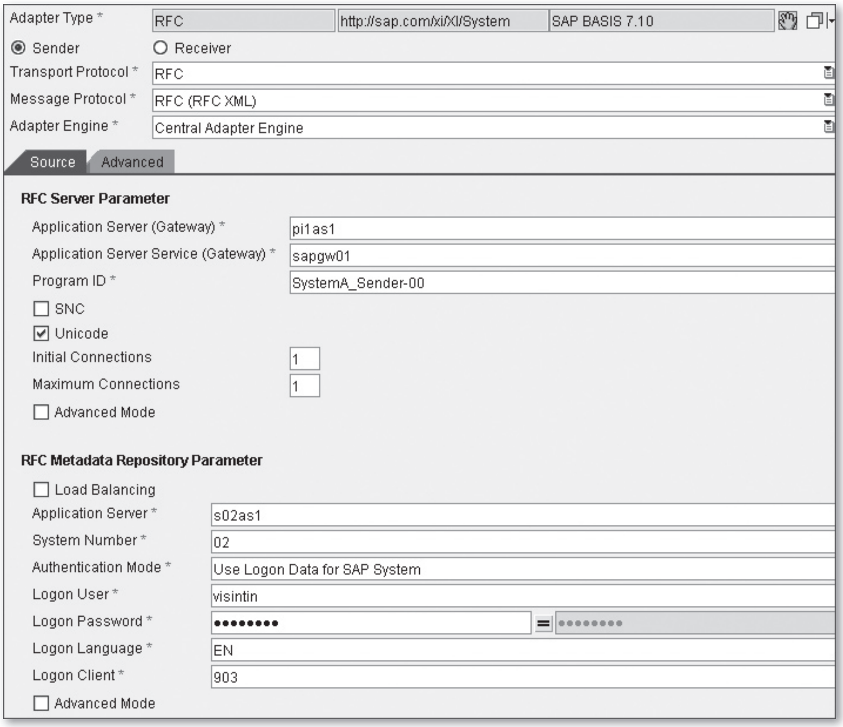
The RFC METADATA REPOSITORY PARAMETER section is used to identify and log on to the system that provides metadata about the RFC interfaces used. This integration is required because the metadata is cross-checked by the PI system when calling the sending RFC adapter. In this example, the RFC interface is imported from system A during the design phase. Enter the APPLICATION SERVER and the SYSTEM NUMBER of system A, and your user `SYS_A-##`, your password, and the corresponding client, before saving the communication channel. If you enable the communication channel at a later stage, the connection test for the destination `SystemA_Sender-##` is carried out successfully from system A.

Figure 4.12 provides an overview of all of the settings for this communication channel.

Creating a file receiver channel

The receiving communication channel for system B is created with the context menu from the Communication Channel path. The name of the new channel should be `File_Receiverchannel_##`. Select system B as the Communication Component.

In the details window, select the File adapter type using the input help and specify the Receiver direction. Set the Transport Protocol to the File System (NFS) parameter, which means that the PI system can use its own local file system to access the directory the file is created in.



Adapter Type *	RFC	http://sap.com/xi/XI/System	SAP BASIS 7.10

(● Sender ○ Receiver)

Transport Protocol *	RFC
Message Protocol *	RFC (RFC XML)
Adapter Engine *	Central Adapter Engine

RFC Server Parameter

Application Server (Gateway) *	pi1as1
Application Server Service (Gateway) *	sapgw01
Program ID *	SystemA_Sender-00
SNC	☐
Unicode	☑
Initial Connections	1
Maximum Connections	1
Advanced Mode	☐

RFC Metadata Repository Parameter

Load Balancing	☐
Application Server *	s02as1
System Number *	02
Authentication Mode *	Use Logon Data for SAP System
Logon User *	visintin
Logon Password *	•••••••• = ••••••••
Logon Language *	EN
Logon Client *	903
Advanced Mode	☐

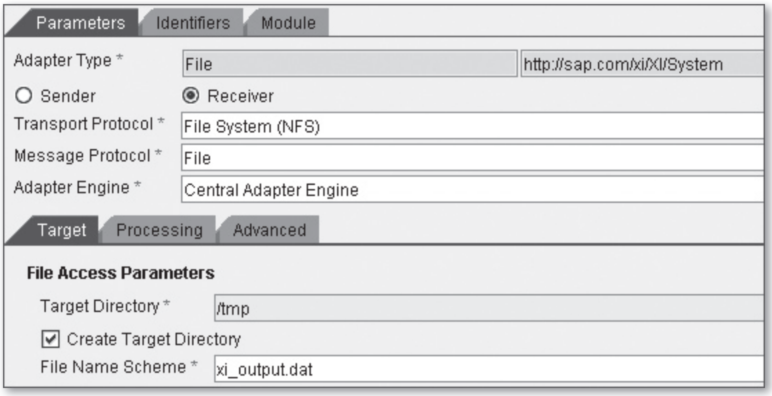
Figure 4.12 Setting Up the RFC Sender Channel for System A

An alternative to the *Network File System* (NFS) is the *File Transfer Protocol* (FTP), which allows access to the file systems of remote computers. If you select FTP, you can specify the server and user data to log on to a remote FTP server. The MESSAGE PROTOCOL field should be set to the FILE value that causes the written file to be stored in PI format. The File Content Conversion characteristic, however, lets you write the file as a list containing several entries.

Setting the source file

The FILE ACCESS PARAMETERS determine the directory to which the file is written, and the scheme for its name. After consulting your landscape administrator, we recommend using */tmp* for Unix installations or *C:\temp* for Windows.

You can choose the FILE NAME SCHEME. However, you should select the name *xi_output_##.dat* for this exercise, where ## represents your participant number. We will refer to this file during the course of this scenario, so, if you select a different name, you need to take this into account in Section 4.1.4, Process and Monitoring (see Figure 4.13).



Adapter Type * File http://sap.com/xi/XI/System
○ Sender ◉ Receiver
Transport Protocol * File System (NFS)
Message Protocol * File
Adapter Engine * Central Adapter Engine

File Access Parameters
Target Directory * /tmp
☑ Create Target Directory
File Name Scheme * xi_output.dat

Figure 4.13 Setting Up the File Receiver Channel for System B

The Processing Parameters on the PROCESSING tab specify how to create the file; that is, if the name scheme specified earlier is used as-is, or if, for example, a time stamp, a counter value, or the message ID should be included in the file name. Select the Directly write mode and the Binary file type. The write mode, Directly, causes data to be written out without using a temporary file. The file type, Binary, makes it so not only text can be output.

In addition to the basic settings, you can also dynamically specify the file storage path by using variable replacement or triggering an operating system command before or after the message processing. Save the receiver channel.

Creating the Connection Elements

Connection elements between the sender and the receiver side

Based on the basics we just created, and the objects in the Enterprise Services Repository, the integration scenario can be completed using some connection elements. The first two missing elements you need to create are the sender and the receiver agreement. They determine how a message is converted from or to the interface of a specific business system so the PI system or the receiving system can further process the message. In the case of the incoming RFC communication channel, for example, the message must be converted from the RFC adapter format to the PI XML format.

Creating a sender agreement

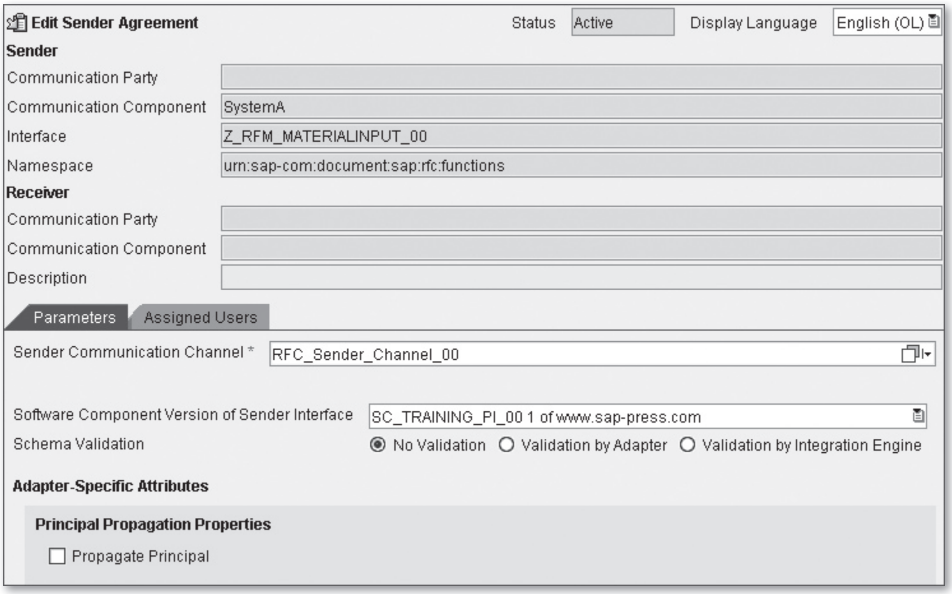
Let's start with the sender agreement, which you can create using the context menu of the SENDER AGREEMENT directory. In the creation dialog, select business system A as the service. The sending interface is the RFC interface `Z_RFM_MATERIALINPUT_##`, which you imported to the Enterprise Services Repository. In the details window of the new object, you can specify the communication channel of the sender by opening the input help and selecting the sender channel `RFC_Senderchannel_##` (see Figure 4.14). Save the sender agreement.

Creating a receiver agreement

As you did for the sender agreement, create a receiver agreement for business system B and the receiving interface `SI_Material_Async_In`. Note that you also need to specify the sending business system A. In the details window, select the channel `File_Receiverchannel_##` as the communication channel of the receiver and save the agreement.

Creating a receiver determination

For logical routing, messages in the PI system first need a receiver determination, which specifies available receiver services for a business system and interface pair. Create a new receiver determination with the corresponding context menu for the sending business system A and the interface Z_RFM_MATERIALINPUT_##.



Edit Sender Agreement — Status: Active — Display Language: English (OL)

Sender

Communication Party	
Communication Component	SystemA
Interface	Z_RFM_MATERIALINPUT_00
Namespace	urn:sap-com:document:sap:rfc:functions

Receiver

Communication Party	
Communication Component	
Description	

Parameters | Assigned Users

Sender Communication Channel *: RFC_Sender_Channel_00

Software Component Version of Sender Interface: SC_TRAINING_PI_00 1 of www.sap-press.com

Schema Validation: [✓ No Validation] Validation by Adapter Validation by Integration Engine

Adapter-Specific Attributes

Principal Propagation Properties

☐ Propagate Principal

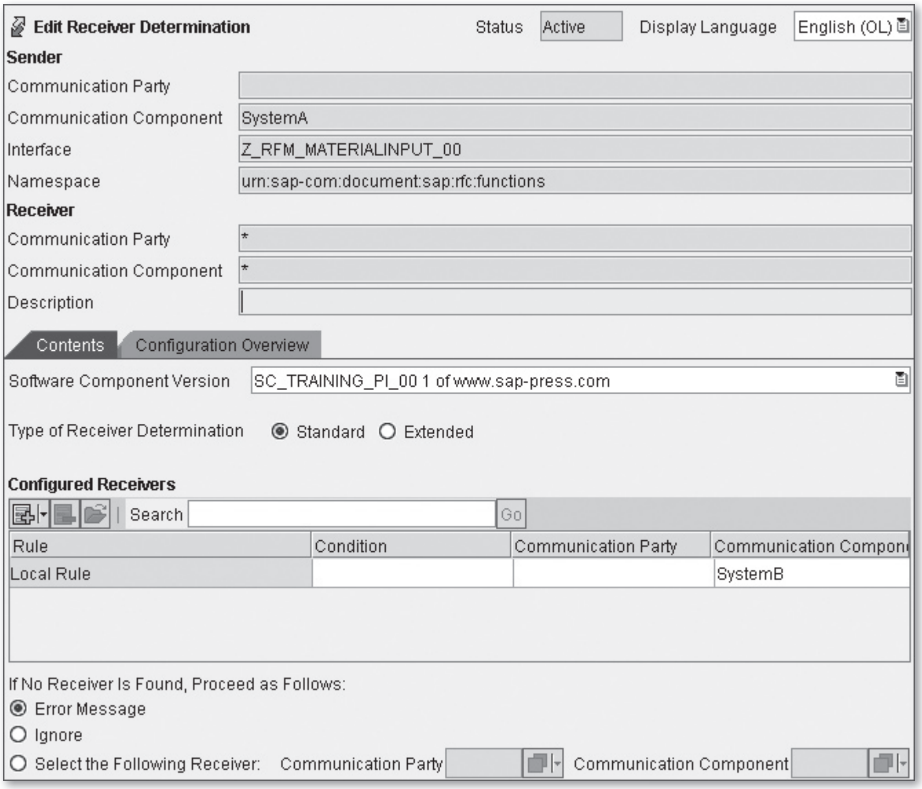
Figure 4.14 Creation of the RFC Sender Agreement

In the details window, the CONFIGURED RECEIVERS area allows you to specify various receivers. If the review result of the relevant condition is true, the message is delivered to this system (see Figure 4.15).

This can also mean that the message is delivered to several systems. For example, the condition can check elements of a message for specific content. If none of the configured systems is specified as the receiver, you can specify a default receiver below the receiver table. In the COMMUNICATION COMPONENT column of the existing row, select business system B as a potential receiver. Because the message in this exercise should always be delivered to this receiver, you don't have to set a condition.

Creating the interface determination

Save the receiver determination and then look at the lower area, CONFIGURATION OVERVIEW, which now includes the SYSTEMB entry. Expand the entry. As you can see, no matching interface determination and no appropriate operation mapping could be determined. Above this listing, click on the New icon to create a new interface determination.


Edit Receiver Determination   Status: Active   Display Language: English (OL)

Sender

Communication Party	
Communication Component	SystemA
Interface	Z_RFM_MATERIALINPUT_00
Namespace	urn:sap-com:document:sap:rfc:functions

Receiver

Communication Party	*
Communication Component	*
Description	

Software Component Version: SC_TRAINING_PI_00 1 of www.sap-press.com

Type of Receiver Determination: [✓ Standard] Extended

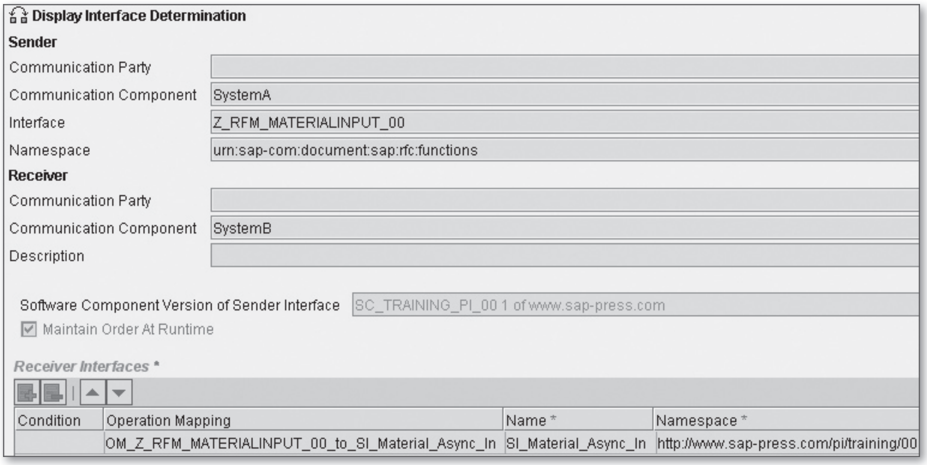
Configured Receivers

Rule	Condition	Communication Party	Communication Compon
Local Rule			SystemB

If No Receiver Is Found, Proceed as Follows:
[✓ Error Message]
Ignore
Select the Following Receiver: Communication Party   Communication Component

Figure 4.15 Creation of the Receiver Determination

By calling the creation dialog from this context, all mandatory fields can be populated; you only need to enter a description. In the details window of the Receiver Interfaces area, use the input help to select your service interface `SI_Material_Async_In` from the namespace. To the left of it, specify the only operation mapping available for the combination of the sending and receiving interfaces (see Figure 4.16). Save and close the interface determination and return to the receiver determination.



Display Interface Determination

Sender

Communication Party	
Communication Component	SystemA
Interface	Z_RFM_MATERIALINPUT_00
Namespace	urn:sap-com:document:sap:rfc:functions

Receiver

Communication Party	
Communication Component	SystemB
Description	

Software Component Version of Sender Interface: SC_TRAINING_PI_00 1 of www.sap-press.com

☑ Maintain Order At Runtime

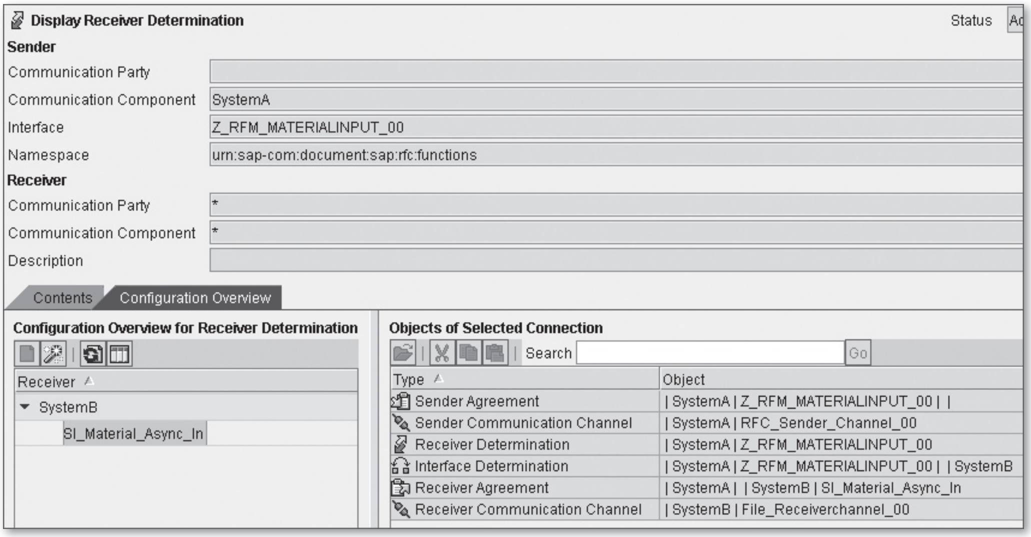
Receiver Interfaces *

Condition	Operation Mapping	Name *	Namespace *
	OM_Z_RFM_MATERIALINPUT_00_to_SI_Material_Async_In	SI_Material_Async_In	http://www.sap-press.com/pi/training/00

Figure 4.16 Editing the Interface Determination for the RFC-to-File Exercise

Activating the new configuration objects

In the lower area, click the REFRESH icon so that the receiver agreement for receiving system B is also displayed with the target interface and the matching operation mapping (see Figure 4.17).



Display Receiver Determination

Status Ac

Sender

Communication Party	
Communication Component	SystemA
Interface	Z_RFM_MATERIALINPUT_00
Namespace	urn:sap-com:document:sap:rfc:functions

Receiver

Communication Party	*
Communication Component	*
Description	

Contents | Configuration Overview

Configuration Overview for Receiver Determination

Receiver
- SystemB
  - SI_Material_Async_In

Objects of Selected Connection

Search | Go

Type	Object
Sender Agreement	| SystemA | Z_RFM_MATERIALINPUT_00 | |
Sender Communication Channel	| SystemA | RFC_Sender_Channel_00
Receiver Determination	| SystemA | Z_RFM_MATERIALINPUT_00
Interface Determination	| SystemA | Z_RFM_MATERIALINPUT_00 | | SystemB
Receiver Agreement	| SystemA | | SystemB | SI_Material_Async_In
Receiver Communication Channel	| SystemB | File_Receiverchannel_00

Figure 4.17 Editing the Receiver Determination for the RFC-to-File Exercise

Save the receiver determination and activate all newly created objects using the Standard Change List in the Change Lists tab. You have now created and activated all objects for this integration scenario.

The folder functionality from the Enterprise Services Repository is also available in the Integration Builder. Here, you create folders by following the OBJECTS • NEW menu path. In the tree on the left, the last point, ADMINISTRATION, contains the FOLDER element. Here, it is possible to create a root folder or assign the new folder to an existing folder as a subfolder (see Figure 4.18). Working with folders in the Integration Directory is the same as in the Enterprise Services Repository.

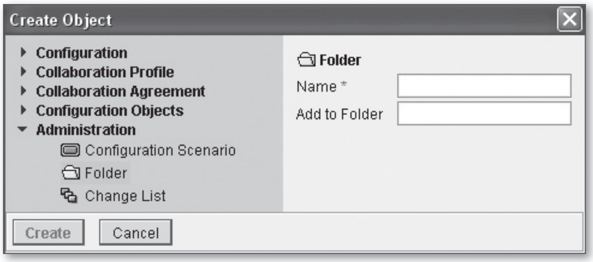


Figure 4.18 Creating a Folder in the Integration Builder

4.1.4 Process and Monitoring

Now that you have created all of the design and configuration objects, you've prepared the integration scenario for the course. Next, you will monitor the process and examine any possible errors.

Course of the Scenario

Calling the ABAP program in system A

Start the configured integration scenario by calling the program Z_PROG_MATERIALINPUT_##. Log in to the client of system A using your user and call Transaction SA38; type the name of the program and execute it.

An input mask for basic material master data is displayed. Enter the data for creating the PI developer manual as a material master record in system B. This material is just used for test purposes; you won't use it to create a production or sales order, for example.

The data corresponds to the mandatory fields of the two views, Basic Data 1 and 2, from Materials Management (MM) in SAP R/3 or SAP ERP Central Component (ECC), respectively. Because this data is used in the second exercise to actually create a material using an IDoc, we recommend using the data from Table 4.4.

Field	Recommended Value
Material	`PI_BOOK-##`
Material description	arbitrary (for example, »SAP PI developer book ##«)
Material type	`FERT` (Finished product)
Industry sector	`1` (Retail)
Material group	`030` (Documentation)
Quantity unit	`ST` (Piece)
Gross weight	arbitrary (for example, `1.2`)
Weight unit	`KGM` (kilogram)
General item category group	`NORM` (Normal item)

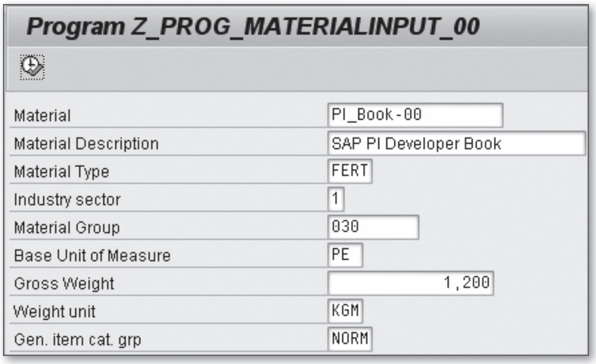
Table 4.4 Recommended Values for Creating a Test Material

Entering the recommended values

This data works in an Internet Demonstration and Evaluation System (IDES) R/3 or ECC system without further adaptation. For the second exercise, you can use the appropriate template files later.

Enter the data in the individual fields and note that the input help displayed for some fields only returns values of the sending system that might not exist in the receiving system (see Figure 4.19).

After you follow the PROGRAM • EXECUTE menu path, or click the corresponding EXECUTE icon, you will receive a success message. This message only notifies you that the function module belonging to the program has been called successfully. However, it does not confirm that the message has been successfully delivered.



Program Z_PROG_MATERIALINPUT_00

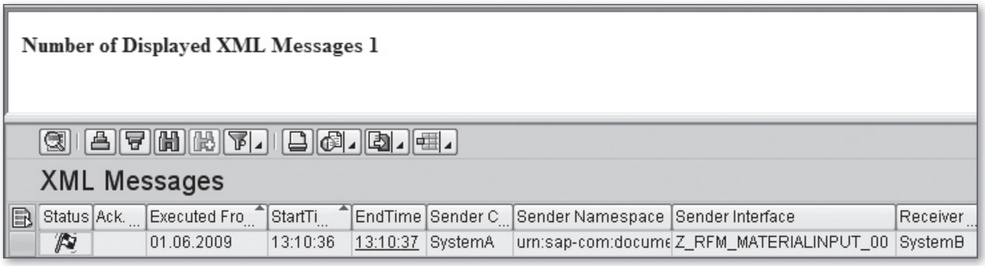
Material	PI_Book-00
Material Description	SAP PI Developer Book
Material Type	FERT
Industry sector	1
Material Group	030
Base Unit of Measure	PE
Gross Weight	1,200
Weight unit	KGM
Gen. item cat. grp	NORM

Figure 4.19 Calling the Program Z_PROG_MATERIALINPUT_##

Monitoring the process

Correct delivery and processing of the message can be verified in the PI system. Log on to the appropriate client and call Transaction SXMB_MONI. Follow the menu path: Integration Engine • Monitoring • Monitor for Processed XML Messages. This opens a selection mask that lets you select all processed messages. If the PI system is used only for training or testing purposes, a restriction is hardly necessary. Otherwise, you could restrict the selection to messages with the sending server SystemA, for example.

Execute the message query via the Program • Execute menu path or the corresponding Execute icon. If your message was successfully delivered and processed, you should see an entry showing a black-and-white checkered flag in the Status column (see Figure 4.20).



Number of Displayed XML Messages 1

XML Messages

Status	Ack. ...	Executed Fro...	StartTi...	EndTime	Sender C...	Sender Namespace	Sender Interface	Receiver ...
🏁		01.06.2009	13:10:36	13:10:37	SystemA	urn:sap-com:docume	Z_RFM_MATERIALINPUT_00	SystemB

Figure 4.20 Display of the First Message in Transaction SXMB_MONI

A green flag means that the message is currently being processed, while a black-and-white checkered flag means the message processed successfully. Most other icons represent an error in our case. You can display the legend of all possible icons via the GOTO • LEGEND menu path, or the corresponding LEGEND icon.

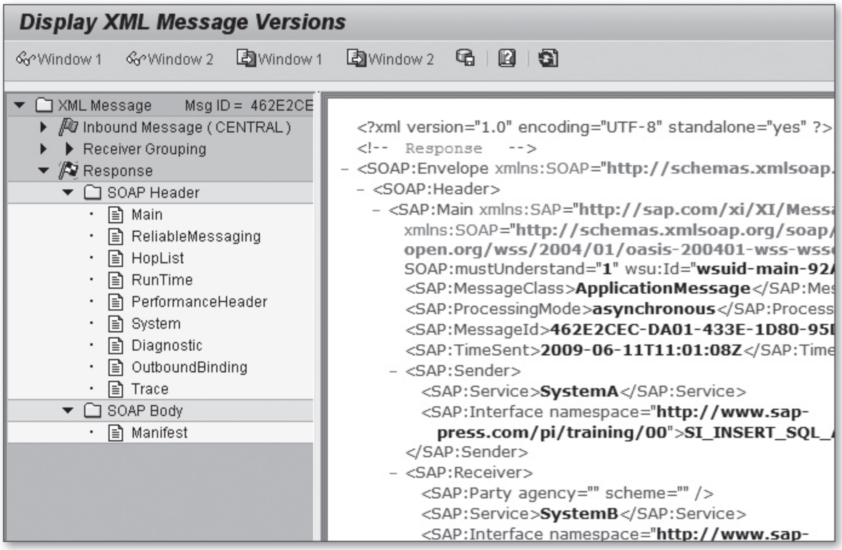
Viewing the created file

To get the ultimate proof that the message was successfully processed, look at the created file. You can do this using Transaction AL11 in the PI system, by clicking on the row of the directory alias `DIR_TEMP`. In the file list, search for a file matching the scheme *pi_output_##.dat*. Double-click on the file to open it. Because the display is limited to a specific width and the lines are not wrapped automatically, we recommend using Transaction ZAPCMD or the file tools provided on the book's website (find it at *http://www.sap-press.com)*.

Troubleshooting in Monitoring

Process analysis

To find the cause of an error in the message display of Transaction SXMB_MONI, double-click in any field of the corresponding row. This brings you to the DISPLAY XML MESSAGE VERSIONS view (see Figure 4.21).



```
XML Message          Msg ID = 462E2CE
  Inbound Message ( CENTRAL )
  Receiver Grouping
  Response
    SOAP Header
      Main
      ReliableMessaging
      HopList
      RunTime
      PerformanceHeader
      System
      Diagnostic
      OutboundBinding
      Trace
    SOAP Body
      Manifest
```

```
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <!--  Response   -->
- <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.
  - <SOAP:Header>
    - <SAP:Main xmlns:SAP="http://sap.com/xi/XI/Mess
        xmlns:SOAP="http://schemas.xmlsoap.org/soap/
        open.org/wss/2004/01/oasis-200401-wss-wss
        SOAP:mustUnderstand="1" wsu:Id="wsuid-main-92
        <SAP:MessageClass>ApplicationMessage</SAP:Me
        <SAP:ProcessingMode>asynchronous</SAP:Process
        <SAP:MessageId>462E2CEC-DA01-433E-1D80-95
        <SAP:TimeSent>2009-06-11T11:01:08Z</SAP:Time
      - <SAP:Sender>
          <SAP:Service>SystemA</SAP:Service>
          <SAP:Interface namespace="http://www.sap-
            press.com/pi/training/00">SI_INSERT_SQL_
        </SAP:Sender>
      - <SAP:Receiver>
          <SAP:Party agency="" scheme="" />
          <SAP:Service>SystemB</SAP:Service>
          <SAP:Interface namespace="http://www.sap-
```

Figure 4.21 Detail View of a Message

In the case of an asynchronous message, you will see the different statuses of the message on its way through the central Integration Engine

(IE). In the directory structure to the left, navigate to the place that has an error icon. In the windows on the right side, look for an error message indicating the cause. In most cases, the error was caused by a mapping or an object that was inadvertently selected from the input help.

Checking the adapters

In some cases, however, the error was caused by incorrectly configured communication channels or adapters. To check this, start Transaction SXMB_IFR, which is used for calling the PI tools. On the bottom right, select the Runtime Workbench and logon using your user `PI-##`.

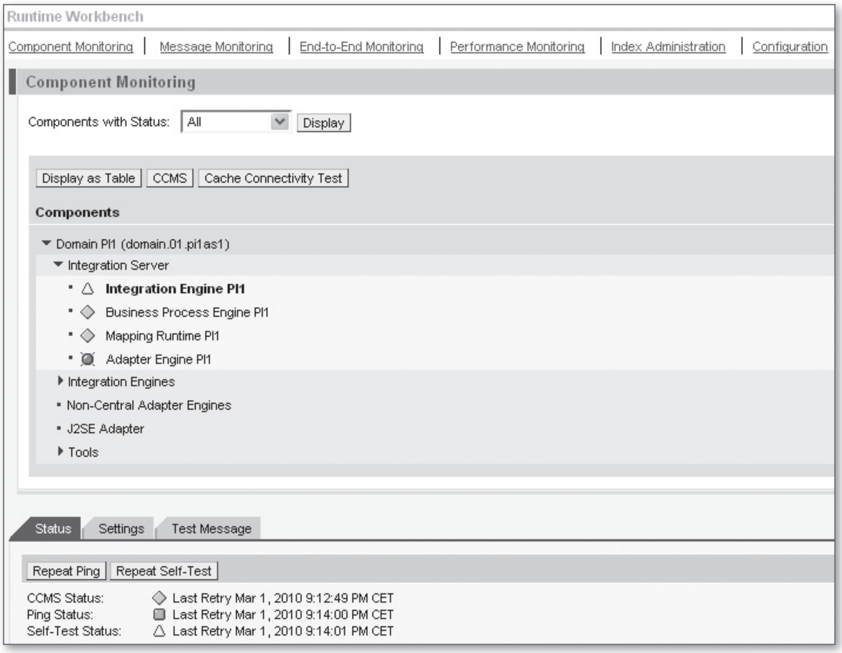
You will see the options for the Runtime Workbench, most of which you will become familiar with while working on the following exercises and the case study. Even though you already know a different way of displaying a message overview using Transaction SXMB_MONI, you can use the MESSAGE MONITORING menu option to view the message status in the PI system. First, select COMPONENT MONITORING, and display the components with every possible status.

In the directory structure of the components, follow the menu path: DOMAIN.XX.<PI-HOSTNAME> • INTEGRATION SERVER • ADAPTER ENGINE. A status view opens beneath the directory structure, providing you with information about the general status of the Adapter Engine. On the top-right, click the ADAPTER MONITORING button (see Figure 4.22).

Selecting the adapter type

After expanding the namespace `http://sap.com/xi/PI/System`, a new browser window opens and displays the selection of all available adapters. A gray diamond next to an adapter type indicates that a communication channel for this type hasn't been created. A green square indicates that all communication channels of this type have been correctly configured and that no error has occurred during processing. A red circle, however, indicates that at least one communication channel of this type is faulty.

You'll need to see if an error occurred for the adapter types RFC or FILE. For a closer analysis, you can click on the relevant type to list all of the communication channels. If the communication channel displays an error, you are presented with a detailed error description to the right, which you can use to correct the error.



Components with Status: All

Components
- Domain PI1 (domain.01.pi1as1)
  - Integration Server
    - Integration Engine PI1
    - Business Process Engine PI1
    - Mapping Runtime PI1
    - Adapter Engine PI1
  - Integration Engines
  - Non-Central Adapter Engines
  - J2SE Adapter
  - Tools

CCMS Status: Last Retry Mar 1, 2010 9:12:49 PM CET
Ping Status: Last Retry Mar 1, 2010 9:14:00 PM CET
Self-Test Status: Last Retry Mar 1, 2010 9:14:01 PM CET

Figure 4.22 Entry Point to Component Monitoring of the Runtime Workbench

4.1.5 Alternative Mapping: ABAP Mapping (Optional)

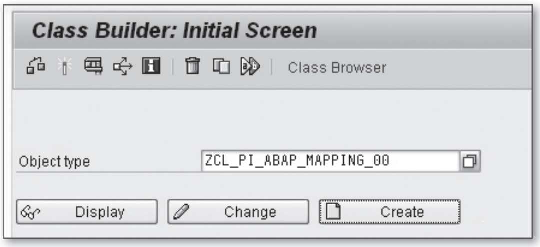
As an alternative to the graphical mapping that you used in this exercise, you will learn how to implement the same mapping using an ABAP class. To do this, you must perform some preparatory steps, and receive authorization for development in the SAP NetWeaver PI system.

Creating the ABAP Mapping

Creation of an ABAP class for the mapping

The ABAP mapping is created as a normal ABAP class in the PI system, and, in operation mapping, is referred to by the class name. As a result, the `Execute` method is called automatically, and does the mapping. This method must be implemented by you; an ABAP mapping cannot be imported into Enterprise Services Repository when developing the ABAP mapping separately.

Log on to the PI system and call the class builder with Transaction SE24. Enter the name of the new class following the schema `ZCL_PI_ABAP_MAPPING_##`, and click CREATE (see Figure 4.23). In the pop-up window, type in a meaningful description for the new class and click Save. If you haven't yet entered a developer key for your user, you will have to do so now.

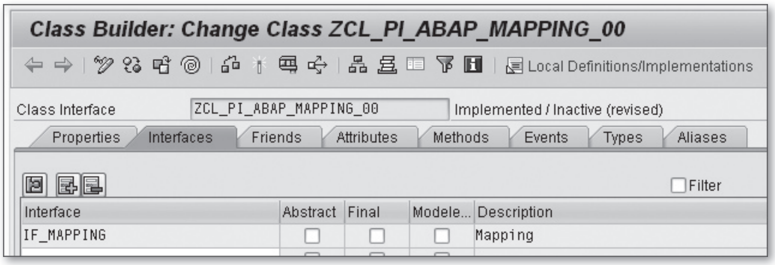


Object type: ZCL_PI_ABAP_MAPPING_00

Figure 4.23 Creating the ABAP Mapping Class in Class Builder

Use of the IF_MAPPING interface

Because the ABAP mapping is a normal ABAP class, you must provide the names of transport requests and of a package for this development (unless you are planning a local development). Once you do this, the class builder will open, and you can see your new ABAP class. Click on the INTERFACES tab and choose the ABAP interface `IF_MAPPING` (see Figure 4.24). This interface contains the `Execute` method with the corresponding signature, and thus represents the definition of the new method.[1]


Class Builder: Change Class ZCL_PI_ABAP_MAPPING_00
Local Definitions/Implementations
Properties
Interfaces
Friends
Attributes
Methods
Events
Types
Aliases
Filter

Class Interface: ZCL_PI_ABAP_MAPPING_00 Implemented / Inactive (revised)

Interface	Abstract	Final	Modele...	Description
IF_MAPPING	☐	☐	☐	Mapping

Figure 4.24 Integration of the IF_MAPPING Interface in the New Class

Target message structure

After integrating the interface, the METHODS tab displays the `Execute` method that you now must implement. To do this, double-click the method name and save the changed class. Before developing the actual coding, let's look at the XML view of the message (see Listing 4.1).

1 You can find detailed information about the interface in the SAP Help Portal at: *http://help.sap.com/saphelp_nwpi711/helpdata/de/ba/e18b1a0fc14f1faf884ae-50cece51b/frameset.htm.*

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:MT_Material
 xmlns:ns1="http://www.sap-press.com/pi/training/##">
     <MATNR>PI_BOOK-##</MATNR>
     <MAKTX>SAP PI Developer Book</MAKTX>
     <ERSDA>01062009</ERSDA>
     <ERNAM>SYS_A-##</ERNAM>
     <MTART>FERT</MTART>
     <MBRSH>1</MBRSH>
     <MATKL>030</MATKL>
     <MEINS>PT</MEINS>
     <BRGEW>1.200</BRGEW>
     <NTGEW>001.080</NTGEW>
     <GEWEI>KGM</GEWEI>
     <MTPOS_MARA>NORM</MTPOS_MARA>
</ns1:MT_Material>
```

Listing 4.1 XML View of a Message Sent after the Mapping

You can access this view in either the file that is created as the result of the first exercise, or by testing the message mapping of the first exercise and displaying the results in the XML view. The only value that is changed in the mapping is the net weight, which is automatically calculated based on the gross weight. The structure and the remaining values are not changed by the mapping, thus keeping the ABAP mapping simple.

Structure of the new method

You can find the detailed source code of the ABAP mapping in Appendix A. Copy the source code, replace the placeholder ##, and activate the method so it can be used later in the operation mapping.

Because a detailed discussion is beyond the scope of this book, we will only briefly discuss the method. First, the iXML library for ABAP Objects is initialized, which allows you to simply parse and reassemble XML documents.[2] Then objects for factories and the input stream are declared. After the definition of the input document, the relevant nodes are declared and extracted from the input document using the `get_elements_by_tag_name` method.

2 Detailed information on the iXML Library of SAP can be found on the SAP Help Portal at: *http://help.sap.com/saphelp_nwpi711/helpdata/de/86/8280ba12d511d5991b00508b6b8b11/frameset.htm.*

After the output document has been declared, it is filled with the values from the input document; most values can be inserted without change. Only the net weight is recreated as a node and calculated on the basis of gross weight. In addition, you must correct the date format from YYYY-MM-DD to DDMMYYYY, so that the file can be transferred into a new material via IDocs (in the next exercise).

After the node assignments to the new document are completed, a custom message is inserted into the trace; this lets you see the details of the execution of the newly-created method in Transaction SXMB_MONI. Finally, the output stream is declared, and a renderer is created that is responsible for compiling the output document.

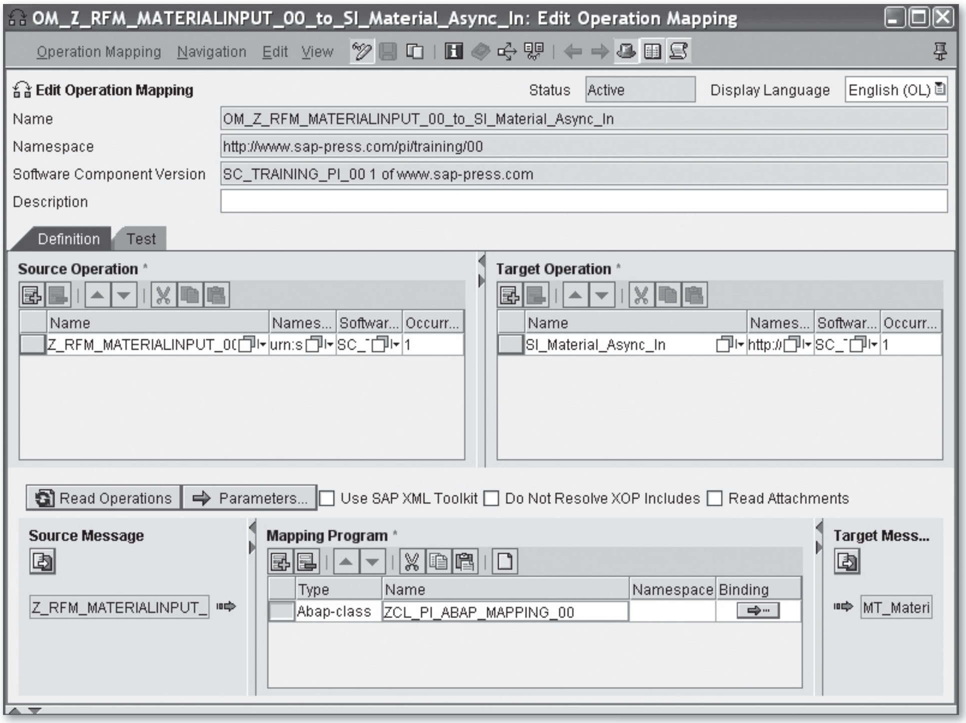
Integrating ABAP Mapping

Adjustment of the operation mapping

After creating and activating the new mapping, you must insert it in the existing operation mapping `OM_Z_RFM_MATERIALINPUT_## to_SI_Material_Async_In`, and activate the object again. To do this, simply open the aforementioned operation mapping, and then switch to change mode. In the bottom-center section of the screen, change the entry in the TYPE column from Message Mapping to `Abap-class`. Then enter the name of the created ABAP class `ZCL_PI_ABAP_MAPPING_##` (see Figure 4.25). The ABAP mapping cannot be tested in the Enterprise Services Repository. Save and activate the change; other changes, such as changes to the configuration, are not necessary.

Testing the changed process

You can run the scenario once again by calling the program `Z_PROG_MATERIALINPUT_##`. The behavior results in the same outcome. By tracing your message in the message monitoring of Transaction SXMB_MONI, you'll find the log entry you specified in the ABAP class (see Figure 4.26).



Status: Active
Display Language: English (OL)
Name: OM_Z_RFM_MATERIALINPUT_00_to_SI_Material_Async_In
Namespace: http://www.sap-press.com/pi/training/00
Software Component Version: SC_TRAINING_PI_00 1 of www.sap-press.com
Description:

Source Operation:

Name	Names...	Softwar...	Occurr...
Z_RFM_MATERIALINPUT_0(	urn:s	SC_	1

Target Operation:

Name	Names...	Softwar...	Occurr...
SI_Material_Async_In	http:/	SC_	1

☐ Use SAP XML Toolkit ☐ Do Not Resolve XOP Includes ☐ Read Attachments

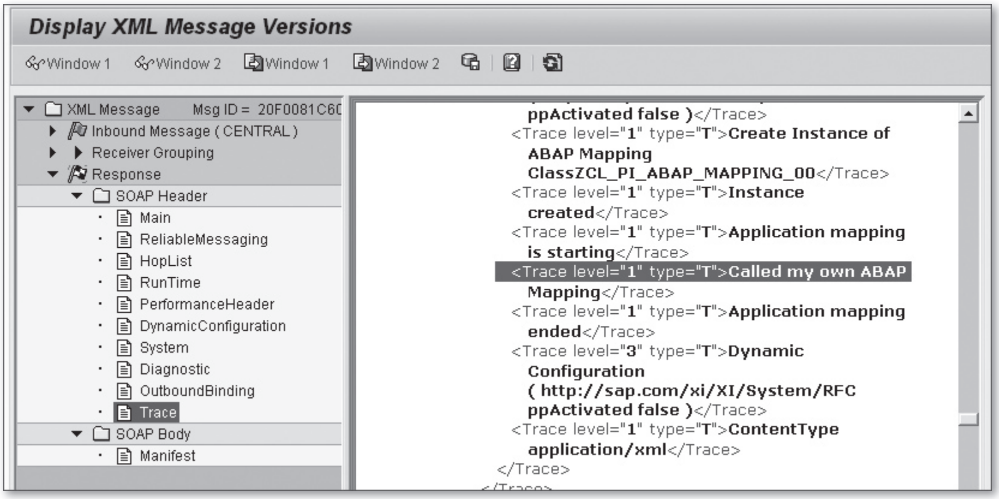
Source Message: Z_RFM_MATERIALINPUT_

Mapping Program:

Type	Name	Namespace	Binding
Abap-class	ZCL_PI_ABAP_MAPPING_00		

Target Message: MT_Materi

Figure 4.25 Integrating ABAP Mapping in Operation Mapping



```
        ppActivated false )</Trace>
      <Trace level="1" type="T">Create Instance of
        ABAP Mapping
        ClassZCL_PI_ABAP_MAPPING_00</Trace>
      <Trace level="1" type="T">Instance
        created</Trace>
      <Trace level="1" type="T">Application mapping
        is starting</Trace>
      <Trace level="1" type="T">Called my own ABAP
        Mapping</Trace>
      <Trace level="1" type="T">Application mapping
        ended</Trace>
      <Trace level="3" type="T">Dynamic
        Configuration
        ( http://sap.com/xi/XI/System/RFC
        ppActivated false )</Trace>
      <Trace level="1" type="T">ContentType
        application/xml</Trace>
    </Trace>
```

Figure 4.26 Trace Entry of the ABAP Mapping in the Message

4.2 Exercise 2: File-to-IDoc

Course of the second exercise

The file containing the material master data that you created in the first exercise now needs to be integrated in business system B to become a material. Although the file has already been transferred to system B from a logical point of view, it technically still resides on the file system of the PI system, in the */tmp* or *C:\temp* directory, respectively. This allows you to keep system A as the sender and system B as the receiver, because system A can also access the file system of the PI system to read the file. System A reads the file using the file adapter and transfers it to the PI system from where the record will be sent as an IDoc to system B.

A diagram of the adapters used and their directions is shown in Figure 4.27.

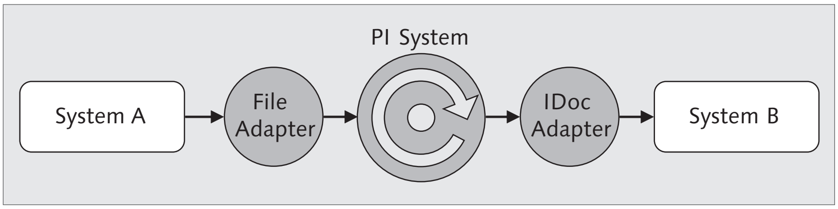


Figure 4.27 Scheme of the Second Exercise File-to-IDoc

Jumping in at Exercise 2

Note

If you didn't do the first exercise, you can find the templates for the file in Appendix A. In this exercise, you will reuse several design elements of the receiving side from the first exercise.

4.2.1 Basics

Overview of the new objects

Now that you have familiarized yourself with the receiver side of the file adapter, it's time to get to know the sender side. The data mapping to an IDoc presents a certain challenge in this exercise, because IDocs contain very sophisticated and complex data structures. Despite the age of this format, it still plays an important role in the SAP environment, partly because of its automatic processing option.

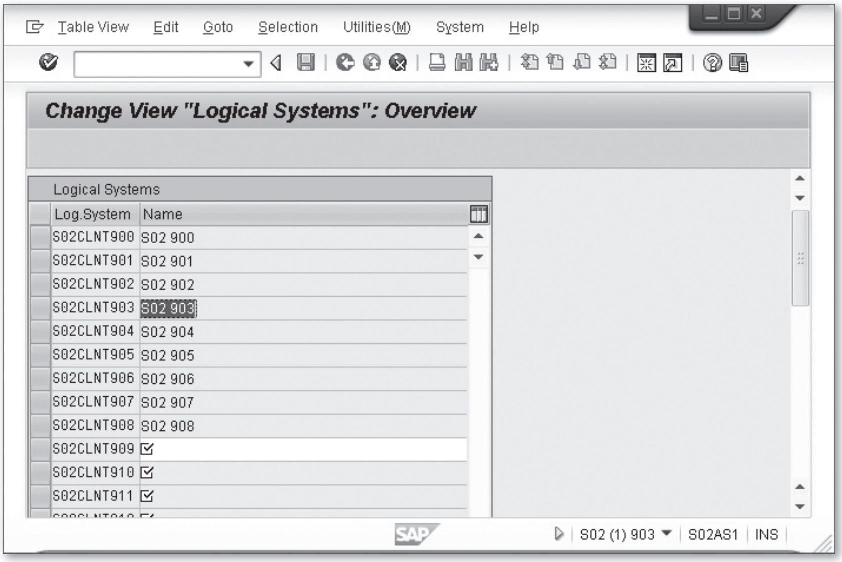
You can take advantage of this in this scenario by creating a partner agreement in system B. This agreement ensures that the incoming material master data is automatically processed in IDoc form (i.e., that the corresponding material is created automatically).

Function of the partner agreement

The partner agreement you will create is not a new element for SAP NetWeaver PI, but an element of the traditional application link enabling (ALE) communication. The partner agreement can only be created once for a sending system (i.e., system A) because the differentiation is made according to the logical system of the sending application. The client of the sending system A, however, can only be assigned a single logical system.[3]

Creating the partner agreement

Log on to system B and call Transaction BD54 to verify that the name of the logical system (system A) is known. Every system must know the names of the logical systems of the IDoc partners. The name of the logical system is usually structured according to the scheme `<SID>CLNT<client>` (see Figure 4.28).



Change View "Logical Systems": Overview

Logical Systems

Log.System	Name
S02CLNT900	S02 900
S02CLNT901	S02 901
S02CLNT902	S02 902
S02CLNT903	S02 903
S02CLNT904	S02 904
S02CLNT905	S02 905
S02CLNT906	S02 906
S02CLNT907	S02 907
S02CLNT908	S02 908
S02CLNT909	✓
S02CLNT910	✓
S02CLNT911	✓

Figure 4.28 Creation of a Logical System

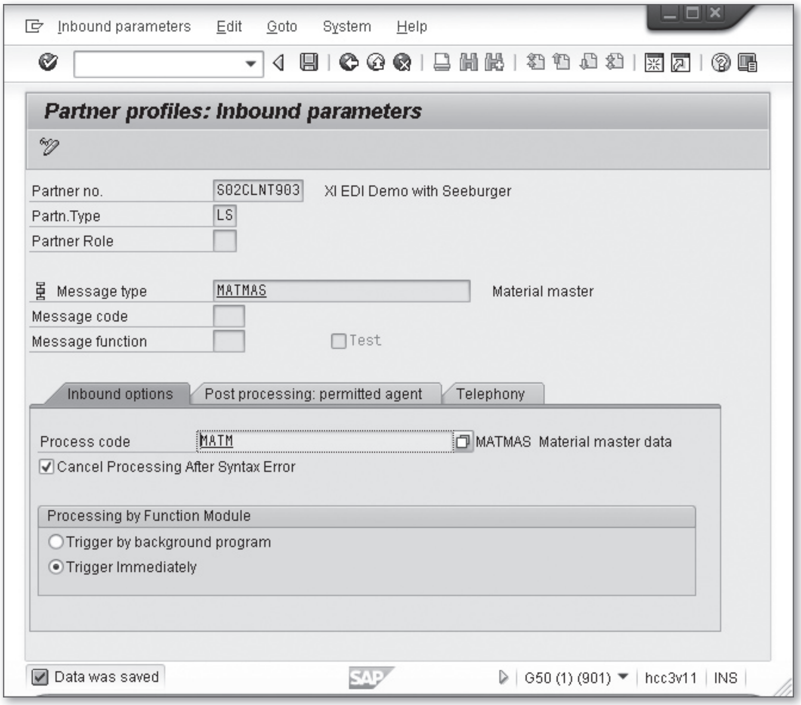
3 If you perform this exercise with other class members, we recommend having the instructor perform the following steps.

Leave Transaction BD54 and call Transaction WE20. The left part of the screen shows the existing partner agreements sorted by partner types. The right side contains detailed information about the selected agreement.

From the menu bar, follow the PARTNERS • CREATE menu path, or click the CREATE icon to create a new partner agreement. Enter the name of the logical system of system A (the one you just checked) as the partner number. Take care to select the partner type LS (logical system) in the details window. Make sure that the partner status value in the Classification tab is set to A, for active.

Configuring the inbound parameters

Next, specify that incoming IDocs of the `MATMAS` (Master Material) type are automatically processed according to a specific pattern. To do this, save the partner agreement and, in the lower area, Inbound parameters, click the Create inbound parameter icon. In the new screen template, select the message type `MATMAS` (see Figure 4.29).



Partner profiles: Inbound parameters

Partner no.	S02CLNT903	XI EDI Demo with Seeburger
Partn.Type	LS	
Partner Role		
Message type	MATMAS	Material master
Message code		
Message function		☐ Test

Process code: MATM — MATMAS Material master data

☑ Cancel Processing After Syntax Error

Processing by Function Module
- ○ Trigger by background program
- ● Trigger Immediately

☑ Data was saved — G50 (1) (901) — hcc3v11 — INS

Figure 4.29 Inbound Parameters of the Partner Agreement in System B

Below the INBOUND OPTIONS tab, select the predefined process code, MATM. If a syntax error occurs, don't terminate the process, as this helps you easily identify potential errors later on. However, if you successfully tested your integration scenario, you can enable the CANCEL PROCESSING AFTER SYNTAX ERROR option by selecting its corresponding checkbox. The PROCESSING BY FUNCTION MODULE option should be performed immediately. Save these settings to exit this transaction.

You just created a partner agreement for communicating with system A via an IDoc.

4.2.2 Design

Overview of the new design objects

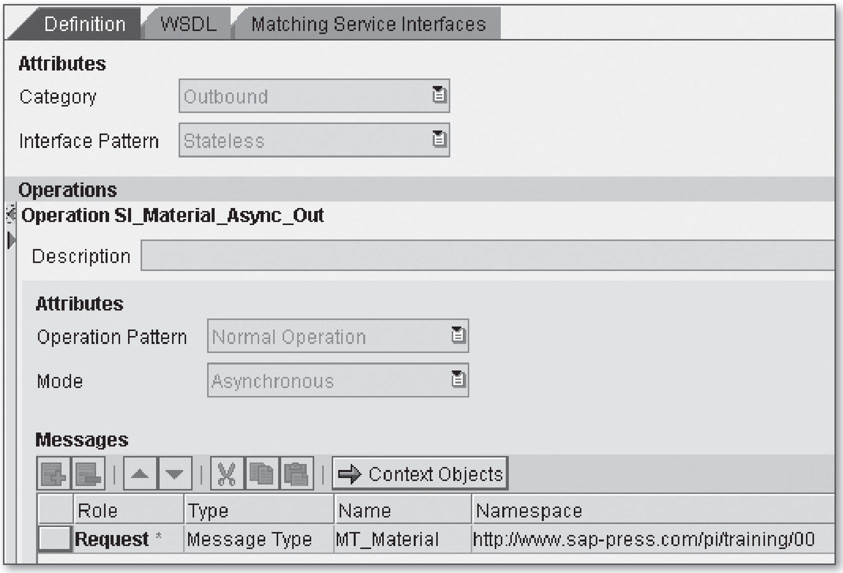
Object creation in the Enterprise Services Repository is much easier after working through the first exercise. If you have already gone through all of the processes, you can reuse some design objects from the first exercise. Reused objects are indicated with an asterisk (*) in Table 4.5.

Object Type	Sender Side	Receiver Side
Service interface	SI_Material_Async_Out	MATMAS.MATMAS02
Message type	MT_Material *	
Data type	DT_Material *	
Operation mapping	OM_SI_Material_Async_Out_to_MATMAS_MATMAS02	
Message mapping	MM_MT_Material_to_MATMAS_MATMAS02	

Table 4.5 Elements in the Enterprise Services Repository for the File-to-IDoc Exercise

Creating the outbound service interface

The sender side, which corresponds to the receiver side from Exercise 1, is already complete, except for the outbound service interface. Open the creation dialog in the SERVICE INTERFACES path, and create the service interface SI_Material_Async_Out. This is taken from the OUTBOUND category, and is used in the ASYNCHRONOUS mode. The output message corresponds to the MT_Material message type already created in the last exercise (see Figure 4.30). Save the completed interface.



Attributes

Category: Outbound

Interface Pattern: Stateless

Operations

Operation SI_Material_Async_Out

Description:

Attributes

Operation Pattern: Normal Operation

Mode: Asynchronous

Messages

Role	Type	Name	Namespace
Request *	Message Type	MT_Material	http://www.sap-press.com/pi/training/00

Figure 4.30 Editing the Outbound Service Interface for the File-to-IDoc Exercise

Importing the IDoc metadata

On the receiver side, you first need to import the metadata of the `MATMAS.MATMAS02` IDoc. You can import the metadata the same way as you imported the RFC interface in Exercise 1. In the Enterprise Services Repository, navigate to your software component version and open the import dialog via the context menu of the Imported Objects directory. If systems A and B run different SAP systems so that the required IDoc type is unknown in system A, you can log on to business system B. Otherwise, you can import from system A.

In the next step, expand the IDocs area, mark the `MATMAS.MATMAS02` type, and import it. Like the interface definition of RFC interfaces, the interface definition of IDocs can be used both as a message type and as a service interface, so more objects are not required on the receiver side.

Creating the message mapping

Create a new message mapping named `MM_MT_Material_to_MATMAS_MATMAS02`, assign the `MT_Material` message type on the sender's side, and assign the `MATMAS.MATMAS02` IDoc type on the receiver's side. Message mapping is a challenge when using IDocs, because the data structure is very complex and can contain several hundred entries. To keep track, Table 4.6 presents the relevant fields of this example.

Data Element of MT_Material/ Constants	Data Element of MATMAS. MATMAS02	Segment of MATMAS. MATMAS02
Constant: 1	BEGIN	None (IDOC)
MT_Material	E1MARAM	None (IDOC)
MT_Material	SEGMENT	E1MARAM
Constant: 005	MSGFN	E1MARAM
MATNR		E1MARAM
Constant: KBG	PSTAT	E1MARAM
Constant: KBG	VPSTA	E1MARAM
Constant: 000	BLANZ	E1MARAM
ERSDA		E1MARAM
ERNAM		E1MARAM
MTART		E1MARAM
MBRSH		E1MARAM
MATKL		E1MARAM
MEINS		E1MARAM
BRGEW		E1MARAM
NTGEW		E1MARAM
GEWEI		E1MARAM
MTPOS_MARA		E1MARAM
MT_Material	E1MAKTM	E1MARAM
MT_Material	SEGMENT	E1MARAM/E1MAKTM
Constant: 005	MSGFN	E1MARAM/E1MAKTM
MAKTX		E1MARAM/E1MAKTM
Constant: D	SPRAS	E1MARAM/E1MAKTM
Constant: DE	SPRAS_ISO	E1MARAM/E1MAKTM

Table 4.6 Assignment of Data Elements in the Message Mapping of the File-to-IDoc Exercise

Structure of the IDoc type

As you can see, element names in the `DT_Material` data type were not chosen by chance. Some fields of the IDoc structure are populated with constants that were not queried in the material input mask.

In particular, pay attention to the `BEGIN` and `SEGMENT` fields, and to the segments that control the creation of the IDoc or its segments, respectively. While `BEGIN` must be assigned a specific value, you can assign any field to the `SEGMENT` fields (and the segments themselves), because this only determines the number of segments in the IDoc. Later in this section, you will work with this more.

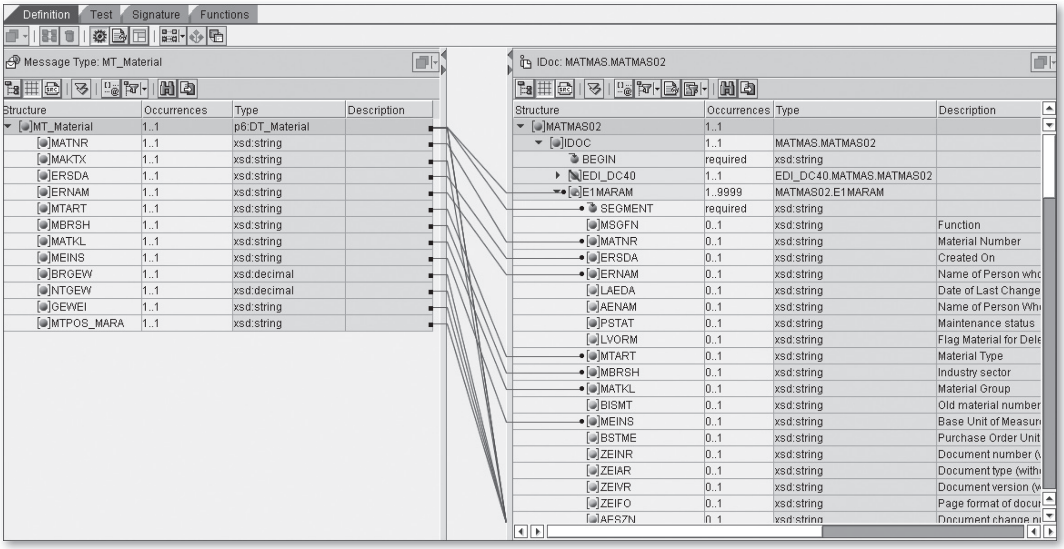
In addition, it's worth mentioning the `MSGFN` fields found in every segment. These fields specify the function to be used by the data in the receiving system. The `005` constant stands for changing or creating.

If you closely examine the structure of the IDoc, you will notice that most segments allow for multiple integrations. Most IDocs are appropriate for mass processing. This IDoc can be used for creating several material master data sets; however, the IDoc in this example is only used to create a single material master record. Documentation of this and all other IDoc types can be found in Transaction WE60.

Deactivating fields

If you examine the table containing the element assignment, you will see that only two of the segments are provided with data. The unused segments can be disabled to prevent them from generating fields. For such a segment, open the context menu and select the Disable Field function. The icon next to the segment is then crossed out and the segment is no longer checked for data. Disable all segments except `E1MARAM` and its child segment, `E1MAKTM`.

Next, create the message mapping (see Figure 4.22) using Table 4.6 and test it. If it runs smoothly, save it. The work area of Figure 4.31 shows how the constant is assigned to the `BEGIN` element in the message mapping.



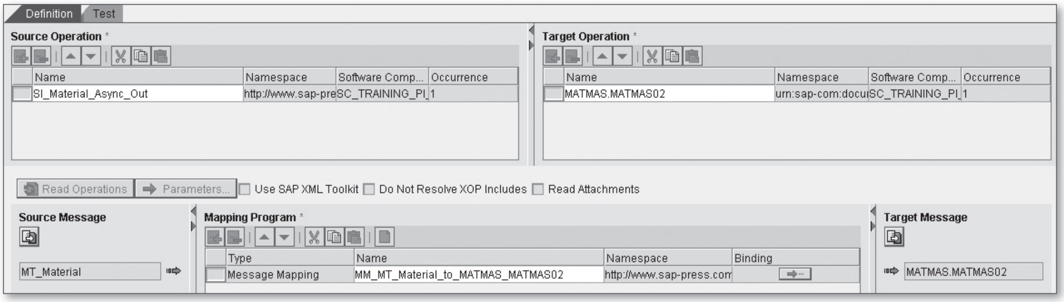
Structure	Occurrences	Type	Description
MT_Material	1..1	p6:DT_Material	
MATNR	1..1	xsd:string	
MAKTX	1..1	xsd:string	
ERSDA	1..1	xsd:string	
ERNAM	1..1	xsd:string	
MTART	1..1	xsd:string	
MBRSH	1..1	xsd:string	
MATKL	1..1	xsd:string	
MEINS	1..1	xsd:string	
BRGEW	1..1	xsd:decimal	
NTGEW	1..1	xsd:decimal	
GEWEI	1..1	xsd:string	
MTPOS_MARA	1..1	xsd:string	

Structure	Occurrences	Type	Description
MATMAS02	1..1		
IDOC	1..1	MATMAS.MATMAS02	
BEGIN	required	xsd:string	
EDI_DC40	1..1	EDI_DC40.MATMAS.MATMAS02	
E1MARAM	1..9999	MATMAS02.E1MARAM	
SEGMENT	required	xsd:string	
MSGFN	0..1	xsd:string	Function
MATNR	0..1	xsd:string	Material Number
ERSDA	0..1	xsd:string	Created On
ERNAM	0..1	xsd:string	Name of Person wh
LAEDA	0..1	xsd:string	Date of Last Change
AENAM	0..1	xsd:string	Name of Person Wh
PSTAT	0..1	xsd:string	Maintenance status
LVORM	0..1	xsd:string	Flag Material for Dele
MTART	0..1	xsd:string	Material Type
MBRSH	0..1	xsd:string	Industry sector
MATKL	0..1	xsd:string	Material Group
BISMT	0..1	xsd:string	Old material number
MEINS	0..1	xsd:string	Base Unit of Measur
BSTME	0..1	xsd:string	Purchase Order Unit
ZEINR	0..1	xsd:string	Document number (
ZEIAR	0..1	xsd:string	Document type (with
ZEIVR	0..1	xsd:string	Document version (w
ZEIFO	0..1	xsd:string	Page format of docu
AESZN	0..1	xsd:string	Document change n

Figure 4.31 Message Mapping of the File-to-IDoc Exercise

Next, create the operation mapping OM_MI_Material_Async_Out_to_MATMAS_MATMAS02 based on the message mapping MM_MT_Material_to_MATMAS_MATMAS02 via the context menu in the appropriate path. Use the two service interfaces SI_Material_Async_Out and MATMAS.MATMAS02, import the interfaces into the lower area of the details window, and assign the message mapping you just created (see Figure 4.32). In the TEST tab, test the operation mapping and save the object.

Creating the operation mapping



Source Operation *

Name	Namespace	Software Comp...	Occurrence
SI_Material_Async_Out	http://www.sap-pre	SC_TRAINING_PI_	1

Target Operation *

Name	Namespace	Software Comp...	Occurrence
MATMAS.MATMAS02	urn:sap-com:docu	SC_TRAINING_PI_	1

☐ Use SAP XML Toolkit ☐ Do Not Resolve XOP Includes ☐ Read Attachments

Source Message: MT_Material

Mapping Program *

Type	Name	Namespace	Binding
Message Mapping	MM_MT_Material_to_MATMAS_MATMAS02	http://www.sap-press.com	

Target Message: MATMAS.MATMAS02

Figure 4.32 Operation Mapping for the File-to-IDoc Exercise

Because this was the last new design object for this exercise, you should now apply all of the changes in the Change Lists tab before switching to the Integration Directory.

4.2.3 Configuration

Overview of configuration objects

Now use the PI tools menu or the direct URL to switch to the Integration Directory. In the previous exercise, you already created the basic elements for the configuration. Because there are hardly any overlaps with the first exercise at the configuration level, you will create all required objects — even the communication channels — and use the Configuration Wizard to create the connection elements. This wizard saves you from several steps and contributes to a smooth implementation.[4] An overview of all of the configuration objects in this exercise is shown in Table 4.7.

Object Type	Sender Side: System A	Receiver Side: System B
Communication channel	`File_SenderChannel_##`	`IDoc_ReceiverChannel`
Sender agreement	`| SystemA	SI_Material_Async_Out	|`	
Receiver agreement		`| SystemA		SystemB	MATMAS.MATMAS02`
Receiver determination	`	SystemA	SI_Material_Async_Out`	
Interface determination	`| SystemA	SI_Material_Async_Out		SystemB`	

Table 4.7 Elements in the Integration Directory for the File-to-IDoc Exercise

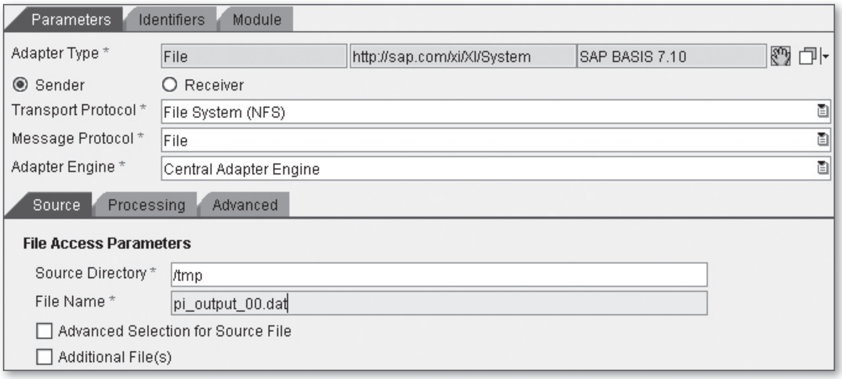
4 This exercise was created on the basis of PI release 7.1 SP7. In this version, it may happen that not all objects are created when generating the configuration objects using the Configuration Wizard. Missing objects must then be created, as described in Section 4.1.3, Configuration. In addition, the value help may not display the right entries within the wizard. In such a case, directly enter the data of the object.

Setting Up the Communication Channels

Creating the file sender channel

To create the sending communication channel for the file adapter, open the creation dialog using the context menu of the Communication Channel object. The new communication channel should be named `File_SenderChannel_##`. Enter system A as the communication component. Enter a description and create the new object.

In the Details window (see Figure 4.33), select the File adapter type and ensure that Sender is selected. Set the Transport Protocol to File System (NFS), because a local directory is accessed. With the Message Protocol, the File setting ensures that the file is read without being converted. The Adapter Engine is the Integration Server (IS).



Adapter Type * File | http://sap.com/xi/XI/System | SAP BASIS 7.10

Sender [✓] Receiver

Transport Protocol * File System (NFS)

Message Protocol * File

Adapter Engine * Central Adapter Engine

File Access Parameters

Source Directory * /tmp

File Name * pi_output_00.dat

☐ Advanced Selection for Source File

☐ Additional File(s)

Figure 4.33 Setting up the File Sender Channel for System A

In the Source Directory field, select the setting you used for the receiver channel in the RFC-to-File exercise (see Section 4.1.3). If you followed the recommendations, this is */tmp* for Unix, or *C:\temp* for Windows. The file name should be *pi_input_##.dat*, where ## represents the participant number.

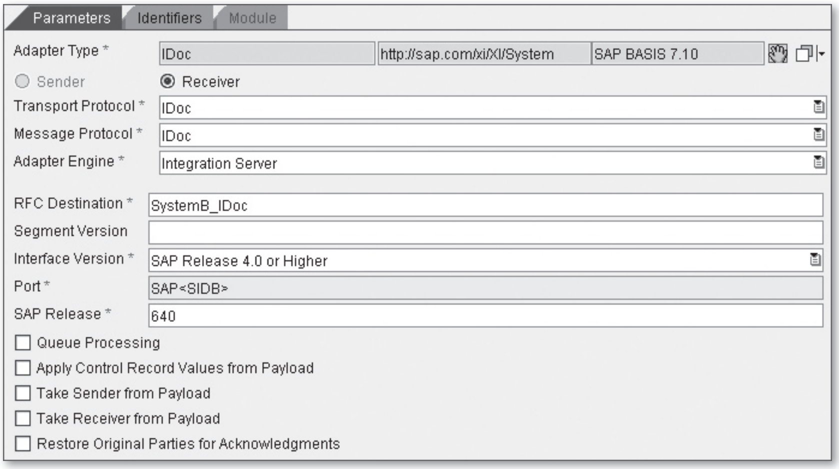
In the Processing tab, you can keep the Quality of Service as Exactly Once. Set the Poll Interval (Sec) to a value between 3 and 10 minutes; otherwise, the load on the IS can be too high. The Retry Interval field specifies the number of seconds the adapter should wait before retrying after a failed read. (This value is not relevant to this exercise.) The Processing Mode of the communication channel can take various characteristics depending on your permissions at the operating system level.

One option is the Archive, which causes the file to be moved to another folder. Alternatively, the file can be deleted or left unchanged in the test operation, which causes the data record to be sent continuously. Use archiving with a time stamp if an appropriate folder is available. The file cannot be archived in the source directory. If you do not have an appropriate folder, you can just delete the file. The processing order can be freely chosen; however, you should set the File Type to Binary.

Creating the IDoc receiver channel

The receiving communication channel for IDocs is only created once (as described in the basics of this exercise) because you cannot distinguish channels between several participants.[5]

Using the context menu of the Communication Channel object, create the communication channel `IDoc_ReceiverChannel` and enter the description. Set the ADAPTER TYPE to IDOC, and set the direction to RECEIVER. The TRANSPORT PROTOCOL, MESSAGE PROTOCOL, and ADAPTER ENGINE fields don't normally offer options, so just ignore them (see Figure 4.34).



Adapter Type * IDoc | http://sap.com/xi/XI/System | SAP BASIS 7.10
Sender [✓ Receiver]
Transport Protocol * IDoc
Message Protocol * IDoc
Adapter Engine * Integration Server
RFC Destination * SystemB_IDoc
Segment Version
Interface Version * SAP Release 4.0 or Higher
Port * SAP<SIDB>
SAP Release * 640
☐ Queue Processing
☐ Apply Control Record Values from Payload
☐ Take Sender from Payload
☐ Take Receiver from Payload
☐ Restore Original Parties for Acknowledgments

Figure 4.34 IDoc Receiver Channel for System B

5 The IDoc communication channel can only be created once by the instructor.

For the RFC Destination, select the `SystemB_IDoc` connection, which you created while preparing for the exercises. The Segment Version is used for integrating with legacy SAP systems, and can be used for sending only those segments that already existed in a specific R/3 release. The Interface Version field (which is mandatory) serves the same purpose but refers to the entire interface definition. For this field, select the SAP Release 4.0 or Higher option. Enter the Port for system B, which was created according to the `SAP<SIDB>` scheme during the preparation (the system ID (SID) for system B is SIDB). In the SAP Release field, enter the release of the receiving SAP system. In the case of SAP ECC 6.0, the entry is 700. Save the object and activate the two new communication channels.

Checking the file adapter

To ensure that the file communication channel has been configured correctly, you can view the adapter in Component Monitoring. Open the Runtime Workbench and display all components by clicking on the entry on Component Monitoring. In the list, click on the Adapter Engine. A new area appears below the list, which contains the Adapter Monitoring button on the right side. Select the File adapter type and check its status.

If no errors occurred, the status should be green. If an error has occurred, you will find a detailed description of the error status. Some errors only occur during data processing, so you should return to the adapter monitoring if you encounter problems at a later stage.

Creating the Connection Elements

Calling the Configuration Wizard

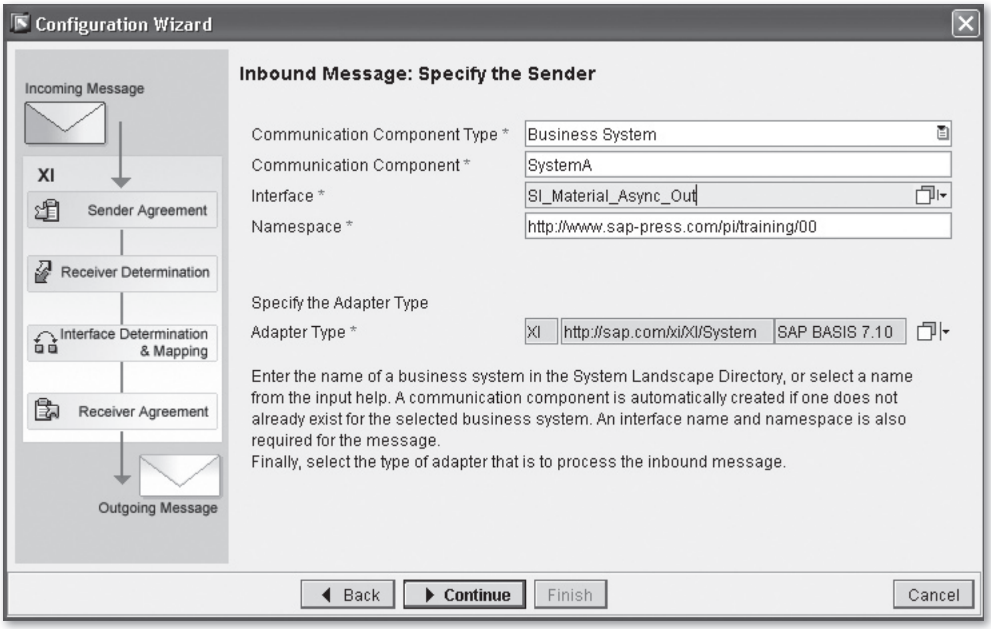
In this exercise, the connection between the two communication channels or between the sender and the receiver interface is created using the Configuration Wizard. The wizard is started via the Tools • Configuration Wizard menu path, or by clicking the corresponding icon in the Integration Directory's toolbar. The wizard collects data for the step-by-step configuration, and creates the appropriate objects. In the progress bar on the left side of the wizard, you can use the orange mark to identify your current position in the configuration process.

Determining the communication type

In the first step, you can select whether to configure an internal or partner communication. Internal communication takes place entirely within the enterprise landscape, while partner communication either entirely or partially involves external business partners. For our scenario, select the Internal Communication option.

Setting the sender parameters

In the next screen (see Figure 4.35), you can specify the values for the sending side. Before you specify other values, however, you should select the sending ADAPTER TYPE; in this exercise, this is the FILE adapter. Next, select business system A as the COMMUNICATION COMPONENT. In the INTERFACE field, specify the interface as `SI_Material_Async_Out`. When selecting the values using the input help, you may need to delete the search criteria and search once more to find a specific interface. If you used the input help for entering the interface, the namespace is populated automatically. Otherwise, specify your namespace as `http://www.sap-press.com/pi/training/##`.



Inbound Message: Specify the Sender

Communication Component Type *	Business System
Communication Component *	SystemA
Interface *	SI_Material_Async_Out
Namespace *	http://www.sap-press.com/pi/training/00

Specify the Adapter Type

Adapter Type * XI | http://sap.com/xi/XI/System | SAP BASIS 7.10

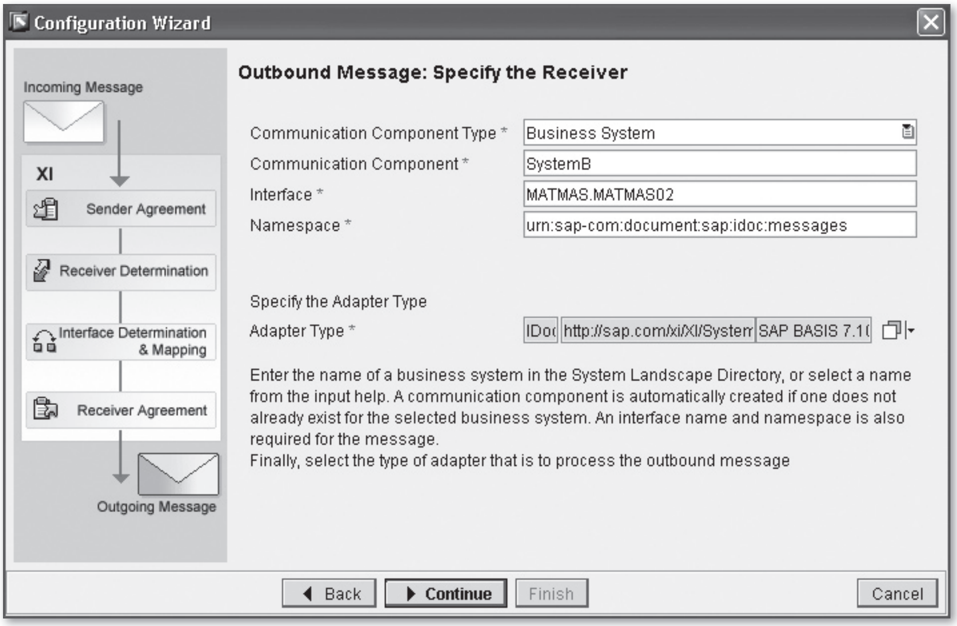
Enter the name of a business system in the System Landscape Directory, or select a name from the input help. A communication component is automatically created if one does not already exist for the selected business system. An interface name and namespace is also required for the message.
Finally, select the type of adapter that is to process the inbound message.

Figure 4.35 Configuration Wizard — Information about the Sender in the File-to-IDoc Exercise

Setting the receiver parameters

When you continue to the next step, you are presented with the same input mask for the receiver side (see Figure 4.36). As before, first select the ADAPTER TYPE IDOC. Set the service to business system B and set the

Interface to `MATMAS.MATMAS02`. By selecting the IDoc adapter type, the namespace is already defined.



Outbound Message: Specify the Receiver

Communication Component Type * Business System
Communication Component * SystemB
Interface * MATMAS.MATMAS02
Namespace * urn:sap-com:document:sap:idoc:messages

Specify the Adapter Type
Adapter Type * IDoc | http://sap.com/xi/XI/Syster | SAP BASIS 7.1(

Enter the name of a business system in the System Landscape Directory, or select a name from the input help. A communication component is automatically created if one does not already exist for the selected business system. An interface name and namespace is also required for the message.
Finally, select the type of adapter that is to process the outbound message

Figure 4.36 Configuration Wizard — Information about the Receiver in the File-to-IDoc Exercise

Determining the sender channel

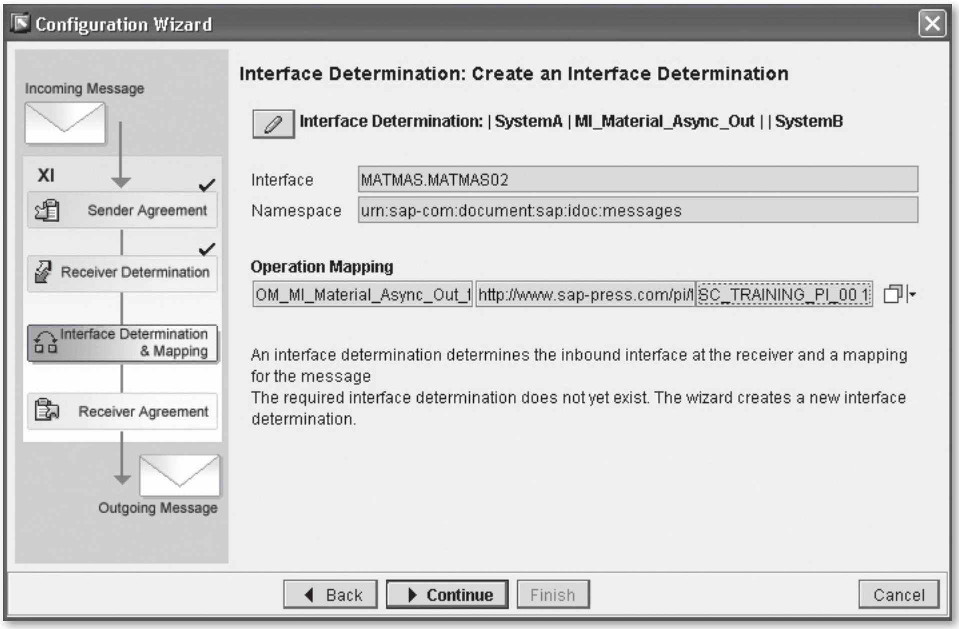
In the next input mask, the communication channel on the sending side is queried to automatically create the sender agreement `|SystemA|SI_Material_Async_Out||`. Specify your sending file communication channel `File_SenderChannel_##` of system A.

Creating the receiver determination

The next step is for creating the receiver determination `|SystemA|SI_Material_Async_Out`. At first, it checks to see if a receiver determination for the sending interface already exists for the relevant system. If the object does exist, it is extended. Otherwise, a new object is created that does not require any input. Because this training scenario shouldn't contain an appropriate receiver determination yet, this step is only for your information.

Creating the interface determination

The next step for creating the interface determination `|SystemA|SI_Material_Async_Out||SystemB` can either be used for information or for checking (see Figure 4.37). Ensure that the MATMAS.MATMAS02 interface has been selected and that the operation mapping is displayed. If the mapping isn't displayed, this usually indicates that it has not been created (or at least not activated).



Interface Determination: Create an Interface Determination

Interface Determination: | SystemA | MI_Material_Async_Out | | SystemB

Interface: MATMAS.MATMAS02
Namespace: urn:sap-com:document:sap:idoc:messages

Operation Mapping: OM_MI_Material_Async_Out_1 | http://www.sap-press.com/pi/ | SC_TRAINING_PI_00 1

An interface determination determines the inbound interface at the receiver and a mapping for the message
The required interface determination does not yet exist. The wizard creates a new interface determination.

Figure 4.37 Configuration Wizard — Information about the Interface Determination of the File-to-IDoc Exercise

Creating the receiver agreement

The last object checked for this scenario is the receiver agreement |SystemA||SystemB|MATMAS.MATMAS02. To create this object, you only need the receiving communication channel IDoc_ReceiverChannel, which should have been entered already. Continue to the last step.

After you have made the necessary changes, you can directly assign the new objects to a specific scenario. Select your scenario (PI_Training_##) and click the FINISH button. The required objects are now created, and reused objects are modified. After the objects have been generated successfully, you will receive a detailed log on the updated and created objects. In release 7.1 SP7, it is possible that not all objects are created. Missing configuration objects must be created manually, as described in

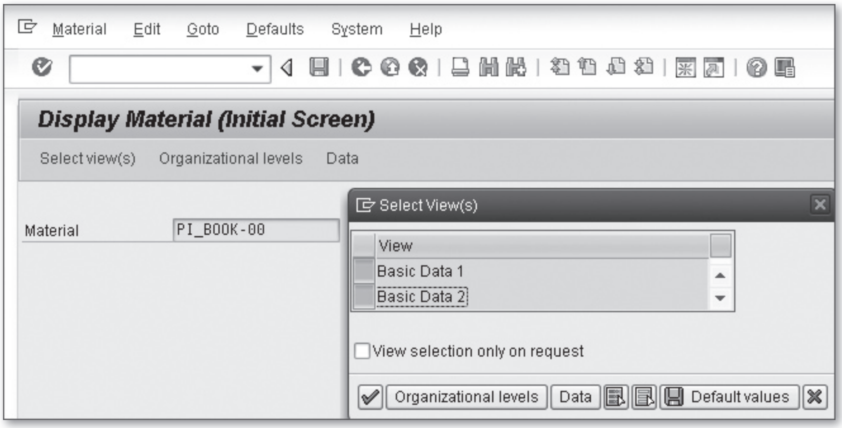
Section 4.1.3. Finally, the changes must be activated in the Change Lists tab.

4.2.4 Process and Monitoring

Controlling the process

The process of the integration scenario in this exercise is triggered by creating the file *pi_input_##.dat* in the directory of the sending file adapter. When creating and placing the file, you can use the template from Appendix A. However, the file should already be in the appropriate directory after processing the first exercise. It may be that the file could not be processed directly after saving the `File-ReceiverChannel`; in this case, call the appropriate program `Z_PROG_MATERIALINPUT_##` from the first exercise again to create a new file.

Depending on the poll interval that has been set in the communication channel, the file is archived or deleted after several minutes. If the process is successful, you can look at your new material, `PI_BOOK-##`, in Transaction MM03 of business system B shortly after the file disappears. In the first screen, select your material and confirm your input by pressing the Enter key. Select the two views, BASIC DATA 1 and BASIC DATA 2, and check the details. To keep the used IDoc structure small, only these two basic views have been created (see Figure 4.38).



Material: PI_BOOK-00

View
Basic Data 1
Basic Data 2

☐ View selection only on request

Figure 4.38 Entry Point for Displaying the Material PI_BOOK-##

Extending the sent data

Using two utilities, you have the option of accessing more views; Transaction WE60 stores the documentation of the entire IDoc type with all of its field names. If you are already familiar with MM in the SAP system, and now want to know which field of the input mask corresponds to a field in the IDoc, you can get this information by pressing F1. For example, open any material using Transaction MM03 and highlight a field. Press the F1 key and click the Technical Information icon. In most cases, the Field Data area contains the exact field name matching that of the IDoc.

Checking the file processing

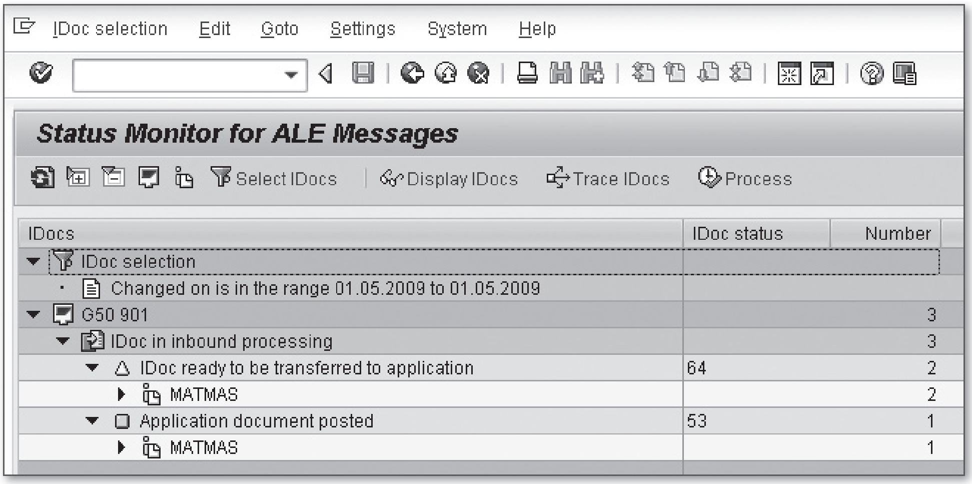
If the material hasn't been created in system B (even after you have waited for a while) you will need to troubleshoot the problem. The first possible source of an error is the file adapter, because it doesn't read and archive the file. In the Component Monitoring of the Runtime Workbench, recheck the status of the corresponding adapter. Some errors are only obvious while a message is being processed.

The next item to troubleshoot would be the message monitoring on the XI system. You can open this via Transaction SXMB_MONI, or the message monitoring in the Runtime Workbench. There you can see if the message has reached the PI system, and analyze whether a content error has occurred for an Integration Builder object. However, if the message shows a black-and-white checkered flag, it has been successfully forwarded to system B.

You can use Transaction SM58 (the monitoring function for transactional RFC calls) to see if there were problems with the technical delivery. For example, one possible error could be a missing or faulty partner agreement created when you prepared the systems for the exercises.

Checking the incoming of the IDoc

If you still haven't discovered the error, the IDoc has been delivered but not processed (i.e., the material has not been created automatically upon receipt of data). To check this, start Transaction BD87 in system B and click the Execute icon without changing the selection criteria. You will receive a list of all IDocs that have been processed on the current day, categorized by various statuses (see Figure 4.39).



IDocs	IDoc status	Number
IDoc selection		
Changed on is in the range 01.05.2009 to 01.05.2009		
G50 901		3
IDoc in inbound processing		3
IDoc ready to be transferred to application	64	2
MATMAS		2
Application document posted	53	1
MATMAS		1

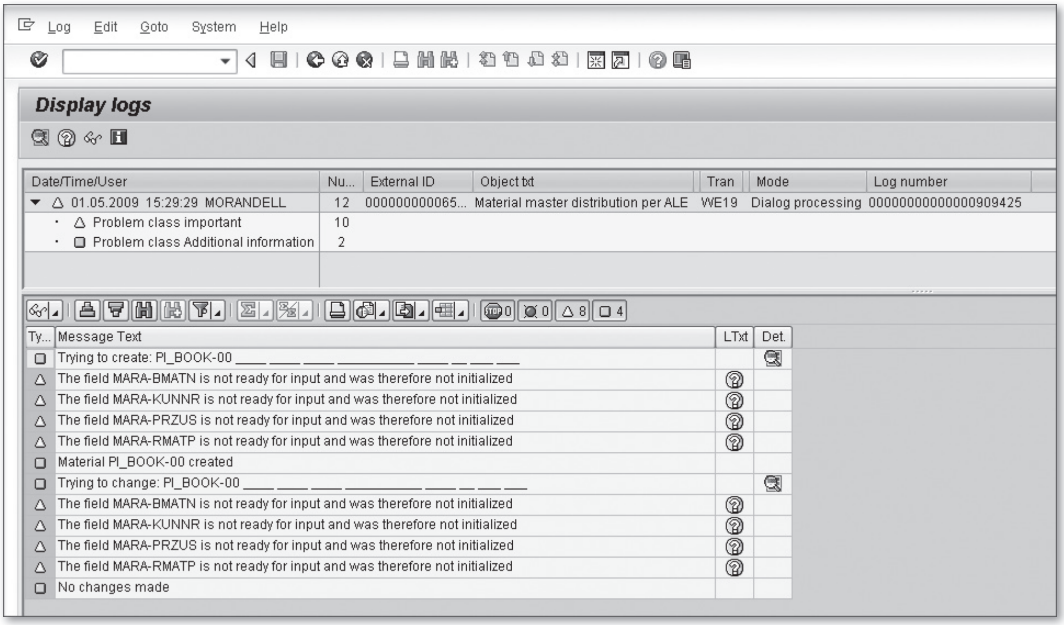
Figure 4.39 Status Monitor for ALE Messages

Navigation in the Status Monitor for ALE messages

To determine the exact source of error when processing a message, click on the appropriate category (e.g., APPLICATION DOCUMENT POSTED). The selection is restricted to the messages in this category. Because system B only differentiates IDocs processing by the names of the logical systems, delivered messages cannot be distinguished by participants.

If you double-click on one of the messages in the list, the corresponding details are displayed. On the left side, expand the Status records directory and the numbers underneath. Every number represents a specific message. In this exercise, you will even find messages for messages that have been successfully processed.

If you double-click on one of these numbers, the status record is displayed. To see a descriptive error message, follow the GOTO • APPLICATION LOG menu path, or click on the appropriate button. You will see warnings and error messages that occurred while the message was processed, and you can see if you have overlooked a field or filled it with the wrong data (see Figure 4.40). IDoc fields are checked, and the input in the relevant transactions.



Date/Time/User	Nu...	External ID	Object txt	Tran	Mode	Log number
▼ △ 01.05.2009 15:29:29 MORANDELL	12	000000000065...	Material master distribution per ALE	WE19	Dialog processing	0000000000000909425
· △ Problem class important	10					
· ▢ Problem class Additional information	2					

Ty...	Message Text	LTxt	Det.
▢	Trying to create: PI_BOOK-00 ____ ____ ____ __________ ____ __ ___ ___		
△	The field MARA-BMATN is not ready for input and was therefore not initialized		
△	The field MARA-KUNNR is not ready for input and was therefore not initialized		
△	The field MARA-PRZUS is not ready for input and was therefore not initialized		
△	The field MARA-RMATP is not ready for input and was therefore not initialized		
▢	Material PI_BOOK-00 created		
▢	Trying to change: PI_BOOK-00 ____ ____ ____ __________ ____ __ ___ ___		
△	The field MARA-BMATN is not ready for input and was therefore not initialized		
△	The field MARA-KUNNR is not ready for input and was therefore not initialized		
△	The field MARA-PRZUS is not ready for input and was therefore not initialized		
△	The field MARA-RMATP is not ready for input and was therefore not initialized		
▢	No changes made		

Figure 4.40 Application Log of an IDoc Message

4.3 Exercise 3: ABAP-Proxy-to-Simple Object Access Protocol (SOAP)

Course of the exercise

In the second exercise, you used an IDoc to import the data you created as a file in the first exercise into business system B. In addition, you could verify that the material master record was properly created. However, in real life, you can't always access both the sending and receiving system.

Therefore, we will implement an integration scenario in this third exercise that allows you to control the successful material creation from system A. You will create an ABAP proxy in system A and call it synchronously. The request is forwarded to a Web service on system B, and a response is returned shortly afterward. The PI system serves as the mediator between the different adapter and data formats.

The scheme of this exercise for synchronous communication is illustrated in Figure 4.41.

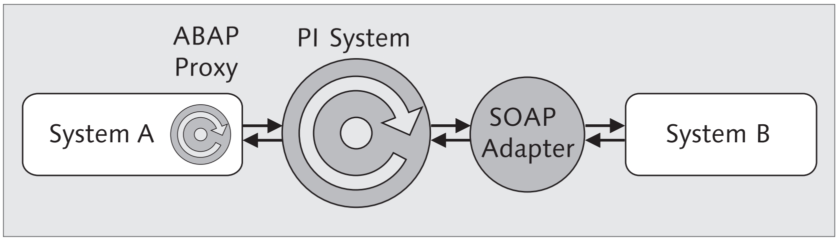


Figure 4.41 Scheme of Exercise 3 ABAP-Proxy-to-SOAP

4.3.1 Basics

Origin of the Web service

The Web service is addressed via the SOAP adapter, and is based on the business application programming interface (BAPI) `BAPI_MATERIAL_EXISTENCECHECK`, which is provided by system B. As of the technical basis of SAP Web Application Server (WAS) 6.20, remote-enabled function modules can be addressed as Web services without specifying any further settings. As with the import of RFC and IDoc interfaces, these Web services can also export the Web Service Description Language (WSDL) description from the SAP system and use it in the PI system.

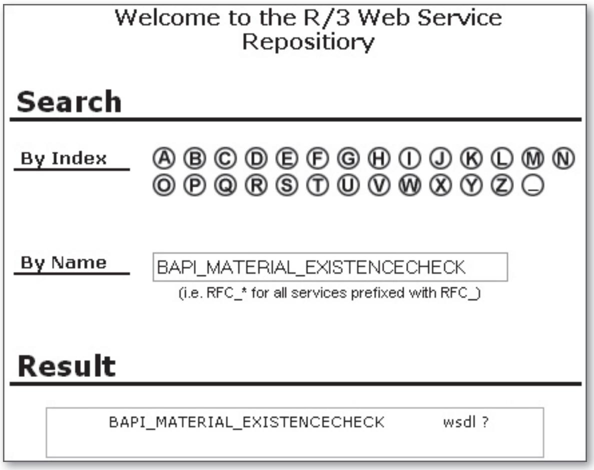
Web Service Repository

To obtain the WSDL description for the appropriate BAPI, use a Service, which you can find via the URL scheme *http://<Host SystemB>:<Port 80XX>/sap/bc/bsp/sap/webservicebrowser/search.html?sap-client=<client>*. The *XX* in the port number represents the instance number of the respective server. The specifications about host and port can be obtained from Transaction SMICM by following the menu path: GOTO • SERVICES.

Open the appropriate URL in the Web browser and log on as user `SYS_B-##`. In the search field, enter the name of the BAPI (`BAPI_MATERIAL_EXISTENCECHECK`), and confirm your selection by pressing the Enter key. The search result shows the BAPI with two links on the right side (see Figure 4.42).

The question mark (?) opens the function module's documentation, if there is any. The WSDL label opens the WSDL description of this function module in a separate window. Because Microsoft Internet Explorer, for example, automatically adds functions for collapsing and expanding individual sections to WSDL descriptions, it is necessary to save the source text of the display. This can be done by following the VIEW •

SOURCECODE menu path. Save the document as *BAPI_MATERIAL_EXISTENCECHECK_SystemB.wsdl*. You will import this file into the PI system during the design phase.


Welcome to the R/3 Web Service Repository

Search

By Index: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z _

By Name: BAPI_MATERIAL_EXISTENCECHECK
(i.e. RFC_* for all services prefixed with RFC_)

Result

BAPI_MATERIAL_EXISTENCECHECK wsdl ?

Figure 4.42 Web Service Repository

The calling ABAP proxy class is created based on a service interface existing in the Enterprise Services Repository. Because inbound and outbound data formats match those of the Web service, you will use the imported WSDL definition as the data and message type of the outbound service interface as well.

4.3.2 Design

Overview of the design objects

In contrast to the previous exercises, you are now facing your first synchronous scenario. This means you not only have to create the design objects to access the Web service, but also for the way back. In this respect, this example is easy because you are using the existing WSDL interface, so you don't have to create any data or message types. In other cases, a synchronous scenario can involve creating up to four different data types and the corresponding message types. While the message mapping is created separately for every direction, the same operation mapping is used for both directions. The new objects to be created for this exercise are listed in Table 4.8.

Object Type	Sender Side	Receiver Side
Service interface	SI_ABAP_PROXY_MAT_EXIST_##_Sync_Out	SI_ws_bapi_material_existencecheck_Sync_In
Message type	ws_bapi_material_existencecheck	
Data type		
Operation mapping	OM_ABAP_PROXY_MAT_EXIST_##_to_ws_bapi_material_existencecheck	
Message mapping (request)	MM_ABAP_PROXY_MAT_EXIST_##_to_ws_bapi_material_existencecheck	
Message mapping (response)	MM_ws_bapi_material_existencecheck_to_ABAP_PROXY_MAT_EXIST_##	

Table 4.8 Elements in the Enterprise Services Repository for the ABAP-Proxy-to-SOAP Exercise

Creating the Interface Objects

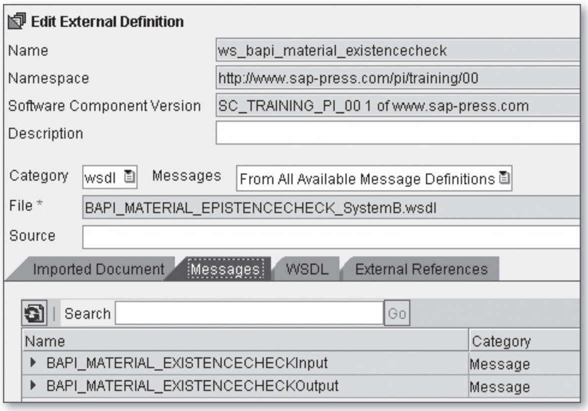
Importing the WSDL description

First, open the context via the Interface Objects • External Definitions menu path in your namespace. Create the external definition `ws_bapi_material_existencecheck`, which will contain the WSDL description of the Web service on system B. In the details window, ensure that the Category wsdl is selected, and, next to the File field, click the Click here to Import external definitions icon. Select the `BAPI_MATERIAL_EXISTENCECHECK_SystemB.wsdl` file on your local machine, which you just transferred from the Web Service Repository (see Figure 4.43).

After the import, you can see the contents of the file in the Imported Document tab. Select the Messages tab to see both of the message types created by the import. Based on the corresponding function module, the Web service contains the definition for the inbound and outbound message. In addition, you can use the message names to determine the connection to the corresponding BAPI.

Creating the synchronous inbound interface

The import of the external definition is handled as a message type with integrated data types. As a result, the service interface must be created manually on the receiving side. Open the creation dialog by choosing the context menu from the Interface Objects • Service Interfaces menu path and create the `SI_ws_bapi_material_existencecheck_Sync_In` service interface.



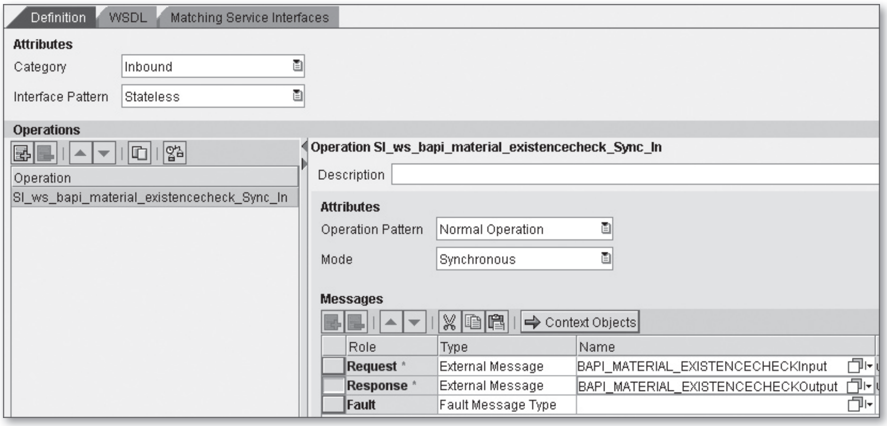
Edit External Definition

Name	ws_bapi_material_existencecheck
Namespace	http://www.sap-press.com/pi/training/00
Software Component Version	SC_TRAINING_PI_00 1 of www.sap-press.com
Description	
Category	wsdl
Messages	From All Available Message Definitions
File *	BAPI_MATERIAL_EPISTENCECHECK_SystemB.wsdl
Source	

Name	Category
▸ BAPI_MATERIAL_EXISTENCECHECKInput	Message
▸ BAPI_MATERIAL_EXISTENCECHECKOutput	Message

Figure 4.43 Importing the External Definition for the ABAP-Proxy-to-SOAP Exercise

Although the Web service processes both inbound and outbound messages due to its synchronous character, it is primarily used as an inbound interface in this scenario. Accordingly, set the INBOUND category and the SYNCHRONOUS mode. This combination gives you the option of specifying both an input and an output message. When selecting a message type from a Web service, you cannot use the drag-and-drop method. Therefore, the easiest option is to use the input help. Below the newly created external definition, select the message type BAPI_MATERIAL_EXISTENCECHECKInput for the input message and the corresponding counterpart for the output message (see Figure 4.44). Save the service interface.



Attributes

Category	Inbound
Interface Pattern	Stateless

Operations

Operation
SI_ws_bapi_material_existencecheck_Sync_In

Operation SI_ws_bapi_material_existencecheck_Sync_In

Description	
Operation Pattern	Normal Operation
Mode	Synchronous

Messages

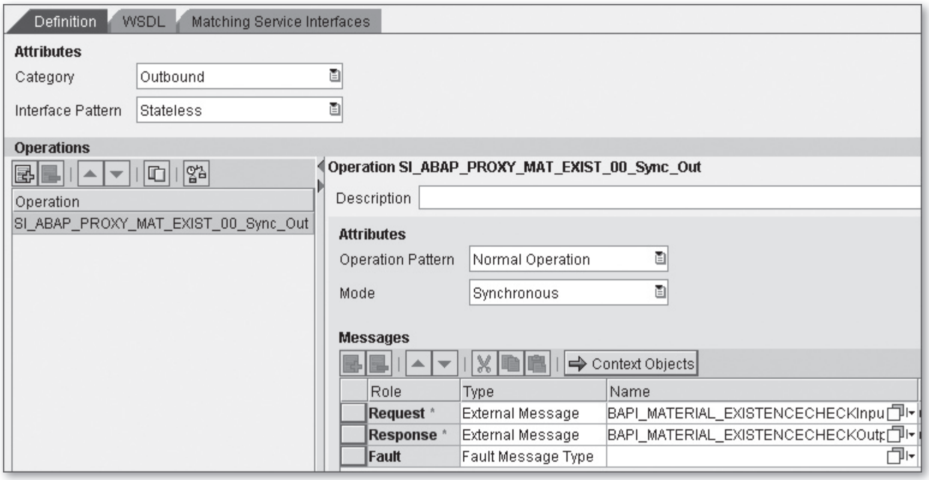
Role	Type	Name
Request *	External Message	BAPI_MATERIAL_EXISTENCECHECKInput
Response *	External Message	BAPI_MATERIAL_EXISTENCECHECKOutput
Fault	Fault Message Type	

Figure 4.44 Creating the Service Interface for the Web Service of the ABAP-Proxy-to-SOAP Exercise

Creating the service interface for the ABAP proxy

Now that the receiver's interface objects have been created, a service interface is still needed for the sender side. The data and the message type are based on the imported WSDL description. Open the creation dialog for service interfaces and create the interface `SI_ABAP_PROXY_MAT_EXIST_##_Sync_Out`. The included participant number is not necessary in the Integration Builder; however, the ABAP classes created later must be distinguishable.

The new interface belongs to the OUTBOUND category and the SYNCHRONOUS mode. For the Request Message, select the `BAPI_MATERIAL_EXISTENCECHECKInput` message type from the external definition, `ws_bapi_material_existencecheck`. The Response Message is of the `BAPI_MATERIAL_EXISTENCECHECKOutput` type from the same external definition (see Figure 4.45). Save the interface and activate all interface objects.



Attributes

Category	Outbound
Interface Pattern	Stateless

Operations

Operation
SI_ABAP_PROXY_MAT_EXIST_00_Sync_Out

Operation SI_ABAP_PROXY_MAT_EXIST_00_Sync_Out

Description

Attributes

Operation Pattern	Normal Operation
Mode	Synchronous

Messages

Role	Type	Name
Request *	External Message	BAPI_MATERIAL_EXISTENCECHECKInpu
Response *	External Message	BAPI_MATERIAL_EXISTENCECHECKOutp
Fault	Fault Message Type	

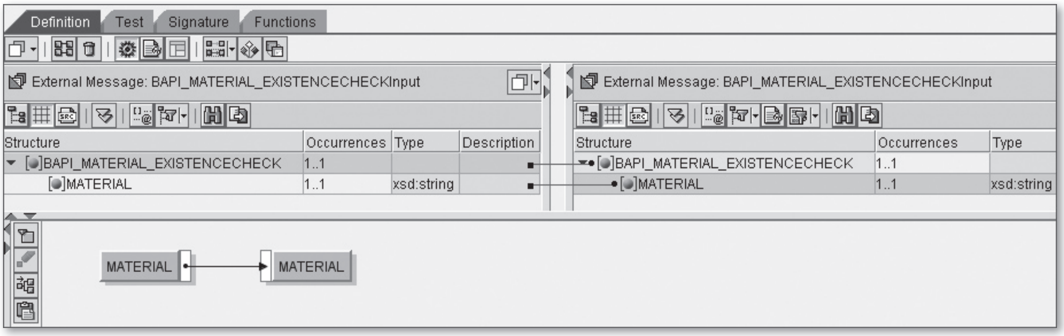
Figure 4.45 Creation of the Service Interface for the ABAP Proxy of the ABAP-Proxy-to-SOAP Exercise

Creating the Mapping Objects

Creating the message mapping for the way to the Web service

Despite previously naming the mapping objects, we won't strictly name the message mappings according to their message types. The reason is that the same message types are used on both the sending and receiving sides, so a standard name doesn't have to be descriptive.

As such, create the message mapping MM_ABAP_PROXY_MAT_EXIST_##_to_ws_bapi_material_existencecheck via the MAPPING OBJECTS • MESSAGE MAPPINGS menu path. This serves as the mapping for the way from the ABAP proxy to the Web service. As an outbound message, select the BAPI_MATERIAL_EXISTENCECHECKInput type from the external definition ws_bapi_material_existencecheck, which can be found in the External Definitions directory. The target message is of the same type. The mapping of the way to the Web service is fairly easy, because only the two MATERIAL fields need to be connected (see Figure 4.46). Test and save the message mapping.



External Message: BAPI_MATERIAL_EXISTENCECHECKInput

Structure	Occurrences	Type	Description
BAPI_MATERIAL_EXISTENCECHECK	1..1		
MATERIAL	1..1	xsd:string	

External Message: BAPI_MATERIAL_EXISTENCECHECKInput

Structure	Occurrences	Type
BAPI_MATERIAL_EXISTENCECHECK	1..1	
MATERIAL	1..1	xsd:string

Figure 4.46 Creating the Message Mapping for the Way to the Web Service in the ABAP-Proxy-to-SOAP Exercise

Creating the message mapping for the way back

The message mapping MM_ws_bapi_material_existencecheck_to_ABAP_PROXY_MAT_EXIST_## for the way back is created according to the same pattern. However, both the outbound and inbound messages are BAPI_MATERIAL_EXISTENCECHECKOutput. The mapping can quickly be implemented by clicking the MAP SELECTED FIELDS AND SUBSTRUCTURES IF NAMES ARE IDENTICAL icon.

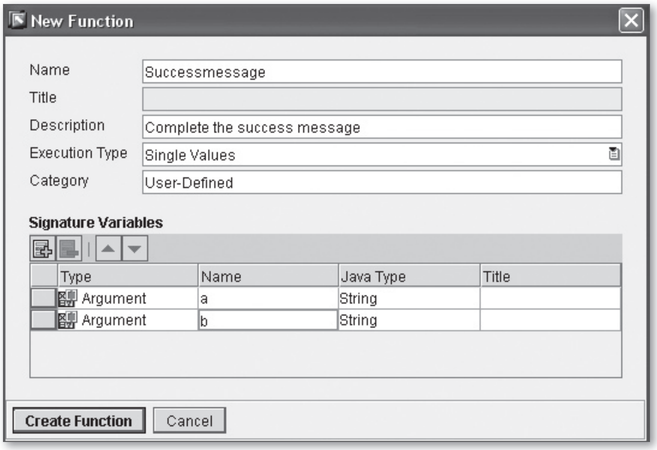
Extending the message mapping by a user-defined function

This simple mapping is used to take the first steps in the area of user-defined mapping programs. For this, you create a user-defined function within the graphical mapping. The function's goal is to fill an element with specific content depending on the contents of another element; the function is an IF construction that could also be implemented using one of the predefined functions. If the Web service detects that the tested material is available, it provides a return code without a descriptive message. If an error occurs, however, a detailed error description is returned. The adaptation of the success and failure responses is achieved using the new function.

One of the affected elements is MESSAGE, which should include a description. The controlling element is NUMBER, which contains a value of 000 in the case of a successful connection.

Creating a user-defined function

To create a user-defined function, double-click on the MESSAGE target element and add the outbound field NUMBER to the work area. In the bottom-left of the work area, click the CREATE NEW FUNCTION icon to create a user-defined Java function. Name the function SUCCESSMESSAGE and enter a description (see Figure 4.47).



Name	Successmessage
Title	
Description	Complete the success message
Execution Type	Single Values
Category	User-Defined

Type	Name	Java Type	Title
Argument	a	String	
Argument	b	String	

Figure 4.47 Creating the User-Defined Function for the ABAP-Proxy-to-SOAP Exercise

Because it is a simple function it is sufficient to only load the values (SINGLE VALUES) in the cache. In the list of arguments, use the appropriate icon to create another argument, b, that, just like argument a, is of the STRING type. The arguments are the transfer parameters to the function.

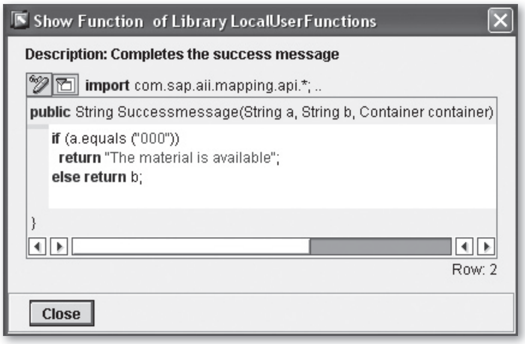
Structure of the source code

You now have the option of creating a Java function that is only valid within this mapping. This procedure should not be confused with the deployment of Java classes, which is referred to as *Java mapping*, and which is described in Section 4.5.5, Alternative Java Mapping (Optional), using an example.

You are still working within the graphical mapping of SAP NetWeaver PI. In the lower part of the work area, the function selection has switched to the USER-DEFINED area, and you can see your new function as a selection option. The newly displayed window represents a small Java editor that already contains a few settings. Insert the source code from Listing 4.2 in the implementation part of the function (see Figure 4.48). You do not have to import any additional classes because the most important classes are imported automatically.

```
if (a.equals (“000”))
  return “The material is available”;
else return b;
```

Listing 4.2 Source Code of the User-Defined Function of the Message Mapping



Description: Completes the success message

```
import com.sap.aii.mapping.api.*; ..
public String Successmessage(String a, String b, Container container)
  if (a.equals ("000"))
   return "The material is available";
  else return b;
}
```

Figure 4.48 User-Defined Function for the ABAP-Proxy-to-SOAP Exercise

Integrating the user-defined function

Save the new function and insert it between the existing NUMBER and MESSAGE (source) and MESSAGE (target) rectangles by clicking on the function name. Drag the connections between the three objects so that NUMBER is connected to the upper white field of the function on the left side. The upper input field of the function is automatically assigned to argument **a**. Connect MESSAGE (source) to the second white field on the left side and assign the function result to the target field (see Figure 4.49).

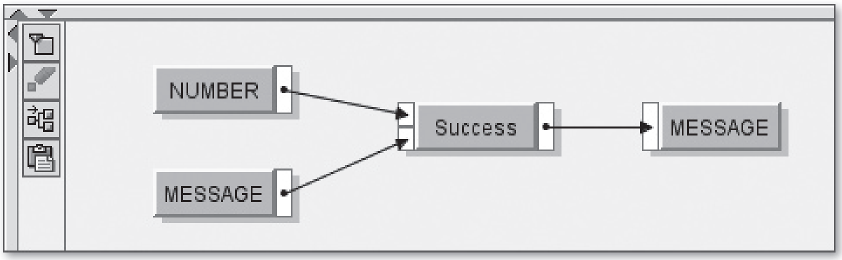


Figure 4.49 Integration of the User-Defined Function for the ABAP-Proxy-to-SOAP Exercise

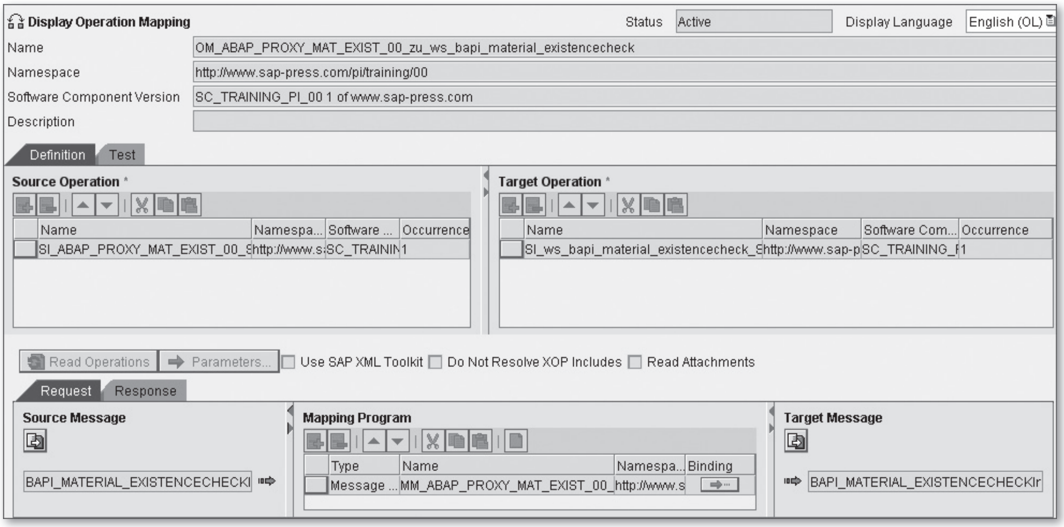
Save and test the entire mapping. Use the value 000 for NUMBER and another value including any message input for testing. If NUMBER equals 000, the message stored in the function should appear in the MESSAGE field.

Creating the synchronous operation mapping

The two message mappings for both directions of the exchange must now be embedded in the operation mapping OM_ABAP_PROXY_MAT_EXIST_##_to_ws_bapi_material_existencecheck. Open the creation dialog for the new operation mapping and enter the name and an appropriate description.

The details window does not show any changes yet compared to an asynchronous communication. For the outbound interface, select the interface SI_ABAP_PROXY_MAT_EXIST_##_Sync_Out using the input help. The target interface is the service interface, SI_ws_bapi_material_existencecheck_Sync_In. In the lower area of the details window, click on the Read Interfaces button and pay attention to the Request label underneath. The simple label turns into two tabs that can be used for configuring each direction (see Figure 4.50).

In the REQUEST tab, select the NAME field in the list of mapping programs, and use the input help to select your mapping. Now click on the RESPONSE tab and repeat this step. In both cases, the input help should only display one entry. Save the operation mapping and activate all mapping objects of this exercise.



Display Operation Mapping

Status: Active

Display Language: English (OL)

Name	OM_ABAP_PROXY_MAT_EXIST_00_zu_ws_bapi_material_existencecheck
Namespace	http://www.sap-press.com/pi/training/00
Software Component Version	SC_TRAINING_PI_00 1 of www.sap-press.com
Description	

Definition | Test

Source Operation *

Name	Namespa...	Software ...	Occurrence
SI_ABAP_PROXY_MAT_EXIST_00_S	http://www.s	SC_TRAININ	1

Target Operation *

Name	Namespace	Software Com...	Occurrence
SI_ws_bapi_material_existencecheck_S	http://www.sap-p	SC_TRAINING_P	1

Read Operations | Parameters... | ☐ Use SAP XML Toolkit | ☐ Do Not Resolve XOP Includes | ☐ Read Attachments

Request | Response

Source Message: BAPI_MATERIAL_EXISTENCECHECK

Mapping Program

Type	Name	Namespa...	Binding
Message ...	MM_ABAP_PROXY_MAT_EXIST_00_	http://www.s	

Target Message: BAPI_MATERIAL_EXISTENCECHECKIr

Figure 4.50 Operation Mapping for the ABAP-Proxy-to-SOAP Exercise

Generating the ABAP Proxy

Providing an ABAP package

The `SI_ABAP_PROXY_MAT_EXIST_##_Sync_Out` service interface created in the Enterprise Services Repository is the basis of proxy generation. An ABAP class is automatically created to allow communication with this interface from an ABAP program. In contrast to remote-enabled function modules, however, no adapter is used. Data is exchanged in the PI format between the local and central IE.

For the remaining steps, you need developer permission in system A, a package `Z_PI_TRAINING`, and a corresponding transport request. The package can be created using Transaction SE80. If you already developed the RFC modules yourself during the first exercise (see Section 4.1, Exercise 1: RFC-to-File) or downloaded the appropriate transport from the website for this book (*http://www.sap-press.com*), you can use the corresponding package.

Generating a proxy class

Log in to system A via the user `SYS_A-##` and call Transaction SPROXY. The structure of this screen is similar to the Enterprise Services Repository: The created software component versions are displayed on the left, with the namespaces listed beneath them. Within the namespaces, however, only the objects of the Interface Objects branch are displayed.

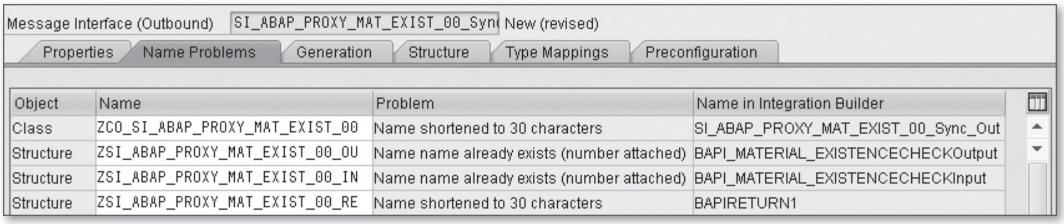
In your namespace, expand the Message Interface (outbound) branch and double-click the service interface SI_ABAP_PROXY_MAT_EXIST_##_Sync_Out. You will be asked if a proxy should be created for this interface; answer Yes. You are then asked for specific settings for the proxy class. Enter the package Z_PI_TRAINING and the prefix Z.

After confirming this information, a warning dialog pops up during the generation process, informing you that there have been name collisions or name shortenings. The reason for this is because the names of the generated ABAP objects may contain a maximum of 30 characters, while the Integration Builder accepts longer names.

Properties of the proxy class

The details window of the newly generated proxy class contains four tabs. The PROPERTIES tab informs you about the classification of the object and the name ZCO_SI_ABAP_PROXY_MAT_EXIST_##. The NAME PROBLEMS tab lists all elements that caused problems during the generation process. The name of the class has been shortened, but is still descriptive and distinguishes the individual participants.

In the right column of this list, you can see which object in the Enterprise Services Repository corresponds to the rows and their contents. The second row displays the structure of the message type BAPI_MATERIAL_EXISTENCECHECKOutput. Change the last two letters to OU so the structure is named ZSI_ABAP_PROXY_MAT_EXIST_##_OU (see Figure 4.51). The counterpart to the message type BAPI_MATERIAL_EXISTENCECHECK-Input should also be renamed to ZSI_ABAP_PROXY_MAT_EXIST_##_IN for consistency. The last structure, which corresponds to BAPIRETURN1, should be renamed to ZSI_ABAP_PROXY_MAT_EXIST_00_RE.



Object	Name	Problem	Name in Integration Builder
Class	ZCO_SI_ABAP_PROXY_MAT_EXIST_00	Name shortened to 30 characters	SI_ABAP_PROXY_MAT_EXIST_00_Sync_Out
Structure	ZSI_ABAP_PROXY_MAT_EXIST_00_OU	Name name already exists (number attached)	BAPI_MATERIAL_EXISTENCECHECKOutput
Structure	ZSI_ABAP_PROXY_MAT_EXIST_00_IN	Name name already exists (number attached)	BAPI_MATERIAL_EXISTENCECHECKInput
Structure	ZSI_ABAP_PROXY_MAT_EXIST_00_RE	Name shortened to 30 characters	BAPIRETURN1

Figure 4.51 Naming Problems when Generating the ABAP Proxy for the ABAP-Proxy-to-SOAP Exercise

Activation of the proxy class

The GENERATION tab displays a list of all of the created objects, regardless of any naming problems. The STRUCTURE tab displays a hierarchical arrangement of all objects. For example, you can see that the new class `ZCO_SI_ABAP_PROXY_MAT_EXIST_##` contains a method called `EXECUTE_SYNCHRONOUS`. Below this method, you can view the parameters from the ABAP program's point of view.

Save the proxy class and specify the corresponding transport request. Check the objects by following the PROXY • CHECK menu path, or by clicking the Check icon. The list of checking messages should contain four yellow warnings, but no red error messages. Confirm the list and activate the objects via the PROXY • ACTIVATE menu path, or by clicking the Activate icon. After you've activated the objects, the activation log can contain warning messages; however, you can ignore the log in this case, because there shouldn't be any errors.

For checking, you can call the Class Builder using Transaction SE24, and display the class `ZCO_SI_ABAP_PROXY_MAT_EXIST_##`. Within the Methods tab, you can see that the method `EXECUTE_SYNCHRONOUS` has been created in addition to the constructor. In addition, the `GET_PROTOCOL` and `GET_TRANSPORT_BINDING` methods are displayed, which have been created but not yet implemented. These two optional methods are not required for this exercise.

4.3.3 Configuration

Overview of configuration objects

Change to the Integration Directory and log in, if necessary. Before you use the Configuration Wizard to configure the objects, you first need to create a communication channel for the receiver side. On the side of the sending system, no communication channel is used, because communication with a proxy does not require an adapter.

The configuration objects you create for this exercise are listed in Table 4.9.

Creating the SOAP communication channel

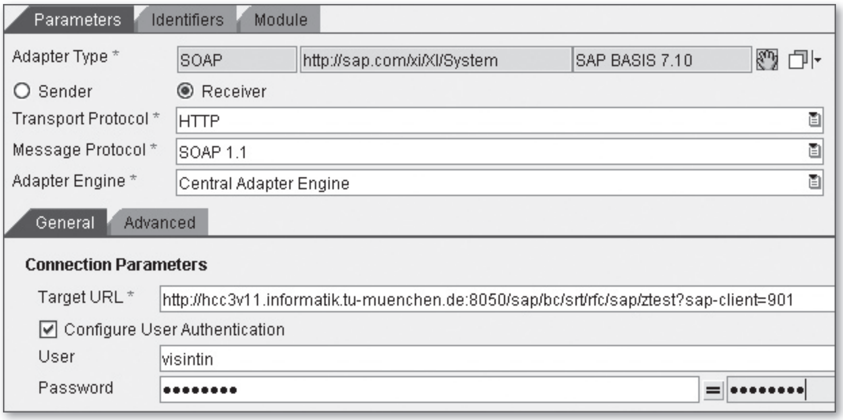
Open the context menu of the Communication Channel path and create a new communication channel for the communication component SystemB. Name it "SOAP_Receiverchannel_##" and enter a description. Continue with Create, and, in the details window, select the SOAP adapter type. The communication channel is created as a RECEIVER. Ensure that the TRANSPORT PROTOCOL is set to HTTP. The other parameters in the header can be left unchanged.

Object Type	Sender Side: System A	Receiver Side: System B
Communication channel		SOAP_ Receiverchannel_##
Receiver agreement		| SystemA | | SystemB | SI_ws_ bapi_material_ existencecheck_ Sync_In
Receiver determination	SystemA | SI_ABAP_PROXY_MAT_EXIST_##_ Sync_Out	
Interface determination	| SystemA | SI_ABAP_PROXY_MAT_EXIST_##_ Sync_Out | | SystemB	

Table 4.9 Elements in the Integration Directory for the ABAP-Proxy-to-SOAP Exercise

In the CONNECTION PARAMETERS area, enter the URL of the target system according to the following scheme: *http://<host SystemB>:<ABAP-Port>/sap/bc/soap/rfc/sap/BAPI_MATERIAL_EXISTENCECHECK?sap-client=<client>*. The client of system B is required to determine against which client user authentication should be performed. To be sure, use Transaction SICF in system B to verify that the corresponding service is active. This service allows you to call remote-enabled function modules as Web services.

In the details window of the communication channel, enable the CONFIGURE USER AUTHENTICATION option. The USER and PASSWORD fields are displayed. Enter the data of your user SYS_B-## (see Figure 4.52), and save the communication channel.



Adapter Type * SOAP | http://sap.com/xi/XI/System | SAP BASIS 7.10

○ Sender ● Receiver

Transport Protocol * HTTP

Message Protocol * SOAP 1.1

Adapter Engine * Central Adapter Engine

Connection Parameters

Target URL * http://hcc3v11.informatik.tu-muenchen.de:8050/sap/bc/srt/rfc/sap/ztest?sap-client=901

☑ Configure User Authentication

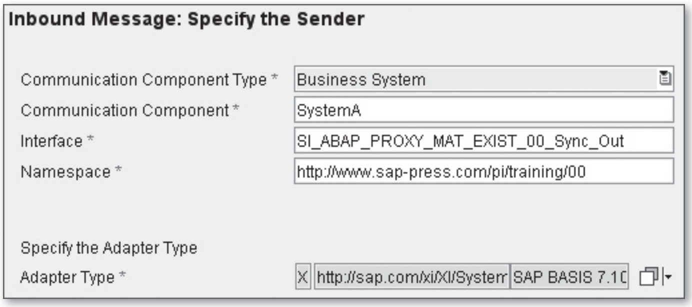
User visintin

Password ••••••••• = •••••••••

Figure 4.52 Setting Up the SOAP Receiver Channel for System B

Using the Configuration Wizard

After the final piece for the configuration has been created, start the Configuration Wizard via the TOOLS • CONFIGURATION WIZARD menu path, or by clicking the CONFIGURATION WIZARD icon. The integration scenario you are implementing belongs to the Internal Communication category. The sender is business system A, which sends data using the service interface `SI_ABAP_PROXY_MAT_EXIST_##_Sync_Out`. The adapter is of the XI type and should already be set (see Figure 4.53).



Inbound Message: Specify the Sender

Communication Component Type * Business System

Communication Component * SystemA

Interface * SI_ABAP_PROXY_MAT_EXIST_00_Sync_Out

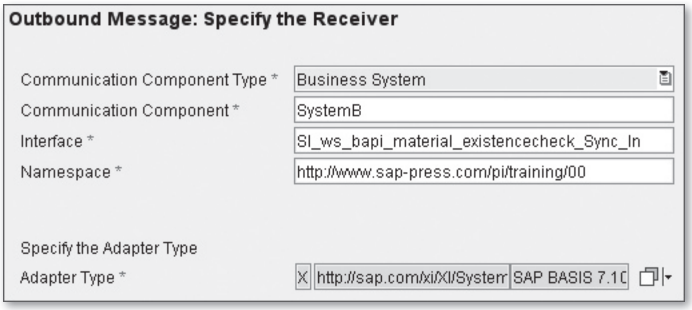
Namespace * http://www.sap-press.com/pi/training/00

Specify the Adapter Type

Adapter Type * X | http://sap.com/xi/XI/Systen | SAP BASIS 7.1(

Figure 4.53 Settings of the Sender in the Configuration Wizard of the ABAP-Proxy-to-SOAP Exercise

The receiver is business system B, which receives information via the SOAP adapter type. The service interface used is `SI_ws_bapi_material_existencecheck_Sync_In` (see Figure 4.54).


Outbound Message: Specify the Receiver

Communication Component Type *	Business System
Communication Component *	SystemB
Interface *	SI_ws_bapi_material_existencecheck_Sync_In
Namespace *	http://www.sap-press.com/pi/training/00

Specify the Adapter Type

Adapter Type *	X http://sap.com/xi/XI/System SAP BASIS 7.10

Figure 4.54 Settings of the Receiver in the Configuration Wizard of the ABAP-Proxy-to-SOAP Exercise

In the next step, you won't need to create a sender agreement, because no conversion takes place. The next screen informs you that the receiver determination SystemA|SI_ABAP_PROXY_MAT_EXIST_##_Sync_Out is created for this scenario. The interface determination shown in the next step should already display the appropriate operation mapping for the target interface SI_ws_bapi_material_existencecheck_Sync_In. Continue with the next step.

The receiver agreement |SystemA||SystemB|SI_ws_bapi_material_existencecheck_Sync_In should refer to the communication channel you just created, SOAP_Receiverchannel_##. Make sure that the new objects are added to your PI_Training_## scenario and finish the wizard. Activate all new and changed configuration objects in your change list.

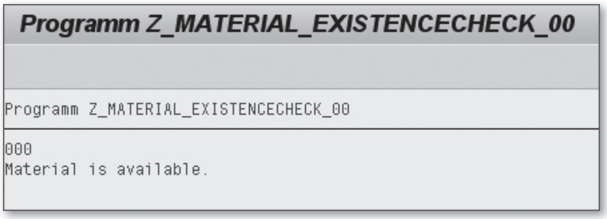
4.3.4 Process and Monitoring

Creating a calling ABAP program

In this scenario, the proxy class with the EXECUTE_SYNCHRONOUS method must be integrated in an ABAP program. Log in to the client of system A as user SYS_A-## and call Transaction SE38 or SE80. You can create the program for calling the proxy method yourself or import the corresponding transport from the website for this book (*http://www.sap-press.com*). If you develop the transport yourself, you will find the sample source code of the Z_MATERIAL_EXISTENCECHECK_## program in Appendix A of this book. You can partially use the data types of the BAPI_MATERIAL_EXISTENCECHECK BAPI, which you can view using the Function Builder in Transaction SE37.

Most variables correspond to the structures that have been generated together with the ABAP class. The ABAP class must be referenced via an object so that the `EXECUTE_SYNCHRONOUS` method can be called. In addition, for the `wa_input` response structure, note that the structure contains `wa_return`, and is therefore detached in a separate step.

Test the program after you have checked and activated it. Specify the `PI_BOOK-##` material created in Exercise 2 and run the program. You should receive the return code 000 and the message "The material is available", as shown in Figure 4.55.



Programm Z_MATERIAL_EXISTENCECHECK_00

```
Programm Z_MATERIAL_EXISTENCECHECK_00

000
Material is available.
```

Figure 4.55 Successful Availability Check of a Material

In contrast to asynchronous scenarios, a synchronous scenario usually doesn't leave any marks. Only asynchronous messages are stored in the persistence layer of SAP NetWeaver PI to enable a later analysis.

4.4 Exercise 4: Business Process Management (BPM)

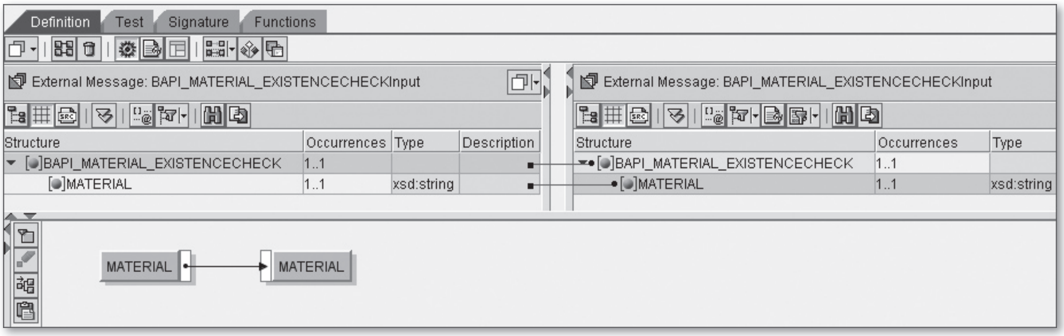
Course of the exercise

The fourth exercise deals with a simple scenario where a department manager or a material manager must be informed about the successful creation of materials. On the sending side, notification is sent using the RFC adapter, while the result is stored as a file in system B.

The important part of this process is that notifications are collected and sorted by creating users. If three messages have been created by one user, those three messages are merged into one and transferred to the material manager. This is controlled using an integration process.

As such, the focus of this exercise does not lie on using new adapters, but on the introduction of SAP NetWeaver PI *cross-component Business Process Management* (ccBPM). You will deal with the corresponding tools in the

Enterprise Services Repository, and the representation in the Integration Directory. Figure 4.56 contains a rough overview of this scenario.



External Message: BAPI_MATERIAL_EXISTENCECHECKInput

Structure	Occurrences	Type	Description
BAPI_MATERIAL_EXISTENCECHECK	1..1		
MATERIAL	1..1	xsd:string	

External Message: BAPI_MATERIAL_EXISTENCECHECKInput

Structure	Occurrences	Type
BAPI_MATERIAL_EXISTENCECHECK	1..1	
MATERIAL	1..1	xsd:string

Figure 4.56 Scheme of Exercise 4 BPM

4.4.1 Basics

Basics of cross-component Business Process Management

ccBPM is mainly about integrating processes that can be implemented within one company or across several different companies. For that purpose, a *Business Process Engine* (BPE) is used to merge individual transformations that were implemented using the Adapter Engine and the IE to a business process.

The *Business Process Execution Language* (BPEL) is used to describe the business processes. Process models are created via a graphical editor that is introduced in the following section. In contrast to the SAP Business Workflow, the BPE communicates with applications on backend systems exclusively using messages.[6] It cannot access processes within applications, the user, or organization management on backend systems. Therefore, the following applies:

- The BPE doesn't control processes within applications. However, you can use messages to integrate applications in cross-system processes.
- The BPE doesn't control user interactions; they can only be controlled on the backend systems.

6 More information about this topic can be found in the SAP Help Portal at: *http://help.sap.com/saphelp_nwpi71/helpdata/EN/3c/831620a4f1044dba38b370f77835cc/frameset.htm*.

The BPM itself does not provide any cross-system monitoring for business documents processed within an integration process. Within the technical monitoring, however, you can display the log of an integration process and the corresponding messages.

Internal procedure of the integration process

Before you start working on this exercise, it's important to look at the steps taking place within the business process. At first, the business process records all messages of a specific interface and sorts them by their creators. A loop is opened for each creator name that collects messages of this type, until three are reached. As soon as three identical messages from the same creator have been collected, the loop ends. These three individual messages are merged into a single message in a new format and then sent. How these considerations are implemented in the BPM of SAP NetWeaver PI is explained in the course of the design phase.

As in Exercise 1, the RFC call starting the integration process is made using the program `Z_PROG_MATERIALINFO_##`, which calls the remote-enabled function module `Z_RFM_MATERIALINFO_##`. The module only works with the material number, the creator name, and the creation date parameters. Again, the source code can be found in Appendix A, and the corresponding transport can be downloaded from this book's web page (*http://www.sap-press.com*).

4.4.2 Design

Before you put the individual objects together to form an integration process, let's first deal with the design objects in this exercise.

Creating the Design Objects

Particular aspects of the design phase

In the context of using business processes, you will get to know a new category of service interfaces: *abstract interfaces*. The important thing to know about abstract interfaces is that they don't have a direction. This means they aren't assigned to the Inbound or Outbound category, and can be used in both directions. The only restriction is that abstract interfaces can only be used in integration processes.

This property requires them to be declared separately as interfaces of the Abstract category, even for imported objects. However, if the abstract interfaces are used for receiving or sending messages within a process, you need a counterpart in the defined direction.

Communication between an abstract and a direction-related interface does not require a mapping, as long as the same message type is used. The mapping objects in this exercise are solely used for generating a single message from the three collected messages; you will deal with the particular aspect of mappings that can have several messages on one side. Because the target format of the integration process needs to be created, the required design objects are listed in Table 4.10.

Object Type	Sender Side	Integration Process/ Receiver
Service interface	SI_RFM_MATINFO_##_Async_Abstract	▶ SI_MatInfo_List_Async_Abstract ▶ SI_MatInfo_List_Async_In
Message type	Z_RFM_MATERIALINFO_##	▶ MT_MatInfo_List
Data type		▶ DT_MatInfo_List ▶ DT_MatInfo
Operation mapping	OM_RFM_MATINFO_##_Async_Abstract_to_MT_MatInfo_List_Async_Abstract	
Message mapping	MM_RFM_MATINFO_##_to_MT_MatInfo_List	
Integration process	IP_MatInfo_##	

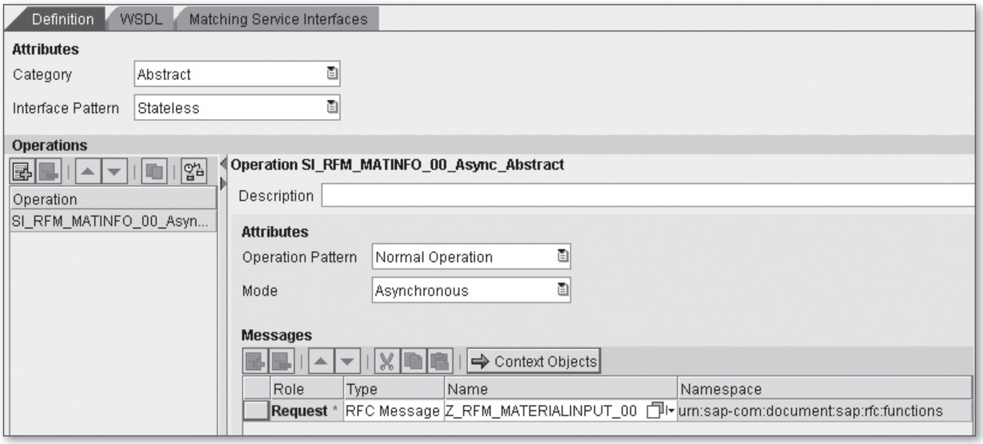
Table 4.10 Elements in the Enterprise Services Repository for the ccBPM Exercise

Creating the objects for the sender side

First, import the definition of the RFC function module Z_RFM_MATERIALINFO_## from business system A.[7] In the Imported Objects directory, use the context menu and log in as user SYS_A-##. Based on the Z_RFM_MATERIALINFO_## message of this imported interface, create the service interface SI_RFM_MATINFO_##_Async_Abstract. Make sure that you set the CATEGORY to ABSTRACT.

7 Importing the interface definition requires that the function module Z_RFM_MATERIALINFO_## exists on system A and that it is remote enabled. This can be done by importing the corresponding transport request, or by manually creating the function module. The source code can be found in Appendix A.

This interface is asynchronous. Choose the STATELESS option for the INTERFACE PATTERN. For OPERATION PATTERN, select the only possible value, NORMAL OPERATION (see Figure 4.57). Save the interface. In doing this, you have created all objects on the sender side.


Definition | WSDL | Matching Service Interfaces

Attributes

Category: Abstract

Interface Pattern: Stateless

Operations

Operation
SI_RFM_MATINFO_00_Asyn...

Operation SI_RFM_MATINFO_00_Async_Abstract

Description:

Attributes

Operation Pattern: Normal Operation

Mode: Asynchronous

Messages

Context Objects

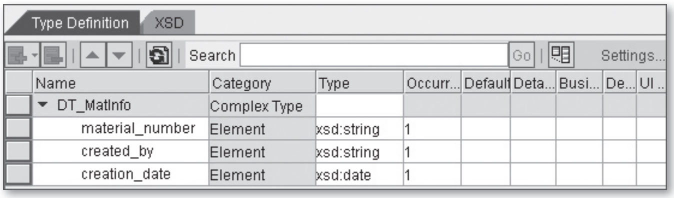
Role	Type	Name	Namespace
Request *	RFC Message	Z_RFM_MATERIALINPUT_00	urn:sap-com:document:sap:rfc:functions

Figure 4.57 Definition of the Abstract Interface SI_RFM_MATINFO_##_Async_ Abstract

Creating the data types

The message type containing the collected notifications at the end of the process is based on a multilevel data type. This means that there is an atomic data type, `DT_MatInfo`, which can contain the contents of one message. To merge several messages into one, the atomic data type must be integrated in a parent type. For this example, the parent data type is `DT_MatInfo_List`.

Start by creating the DT_MatInfo data type, and add three elements containing the material number (`material_number`), the creator (`created_by`), and the creation date (`creation_date`). The first two values are of the `xsd:string` type, while the creation date is of the `xsd:date` type (see Figure 4.58). Save this data type.

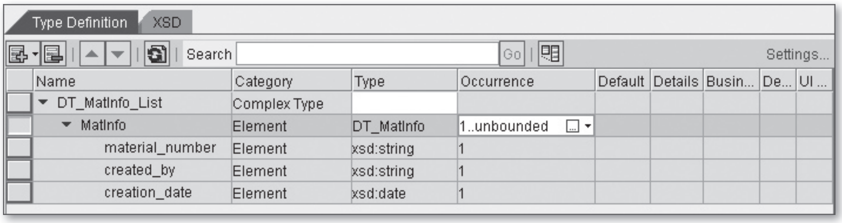


Name	Category	Type	Occurr...	Default	Deta...	Busi...	De...	UI ...
DT_MatInfo	Complex Type							
material_number	Element	xsd:string	1					
created_by	Element	xsd:string	1					
creation_date	Element	xsd:date	1					

Figure 4.58 Structure of the DT_MatInfo Data Type

Next, create the `DT_MatInfo_List` data type so it only contains a single element. This element, `MatInfo`, is of the `DT_MatInfo` type, which you can select using the search help. Once you have created a nested structure, you can record a notification.

To convert the data type into a list, though, you need to change the occurrence of this element. When you double-click in the OCCURRENCE column of the element, a new window opens. Click on Properties; from the input help of the MaxOccurs field, select the unbounded entry so an unlimited number of notifications can be recorded. Confirm your selection and save the data type. The data type should now be structured as shown in Figure 4.59.



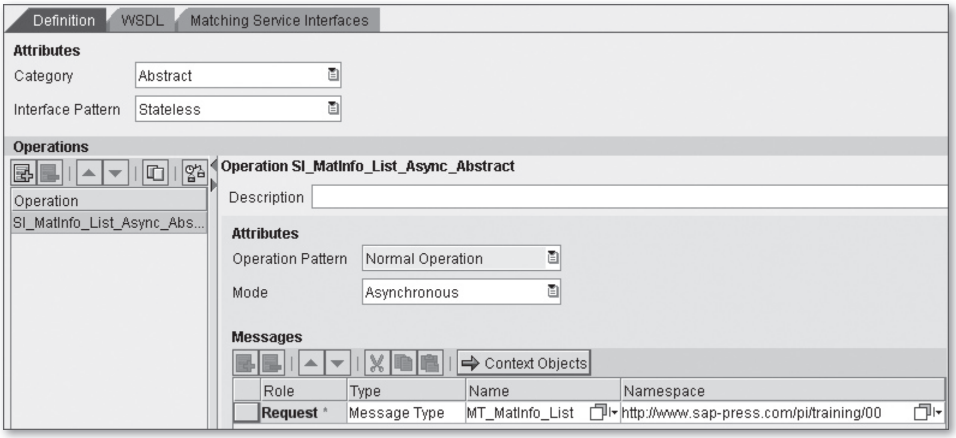
Name	Category	Type	Occurrence	Default	Details	Busin...	De...	UI ...
DT_MatInfo_List	Complex Type							
MatInfo	Element	DT_MatInfo	1..unbounded					
material_number	Element	xsd:string	1					
created_by	Element	xsd:string	1					
creation_date	Element	xsd:date	1					

Figure 4.59 Structure of the DT_MatInfo_List Data Type

Creating the remaining objects on the receiver side

Create the message type `MT_MatInfo_List` and base it on the `DT_MatInfo_List` data type. Integrate this new message type in the service interface `SI_MatInfo_List_Async_Abstract`, which you will use later in the integration process. The service interface belongs to the `Abstract` category and the `Asynchronous` mode. You can leave both pattern fields unchanged (see Figure 4.60). Save the abstract interface.

Sending a message via this interface requires a counterpart in the form of an interface belonging to the Inbound category. Therefore, create a service interface SI_MatInfo_List_Async_In that also uses the MT_MatInfo_List message type (however, this interface is used as an Inbound interface for asynchronous communication). Save this interface object. You have now created all objects for the receiver side.



Attributes

Category: Abstract

Interface Pattern: Stateless

Operations

Operation: SI_MatInfo_List_Async_Abs...

Operation SI_MatInfo_List_Async_Abstract

Description:

Attributes

Operation Pattern: Normal Operation

Mode: Asynchronous

Messages

Role	Type	Name	Namespace
Request *	Message Type	MT_MatInfo_List	http://www.sap-press.com/pi/training/00

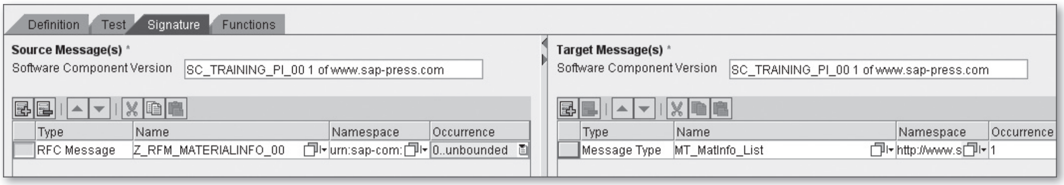
Figure 4.60 Structure of the SI_MatInfo_List_Async_Abstract Service Interface

Creating the mapping objects Mapping isn't required for connecting abstract and direction-related interfaces of the same message type. However, for converting the three notifications into a single message, a mapping is required at the message and interface level.

Start by creating the message mapping MM_RFM_MATINFO_##_to_MT_MatInfo_List. The outbound message is Z_RFM_MATERIALINFO_## of the RFC interface of the same name. The target message is the MT_MatInfo_List type you just created.

As you can see, the mapping does not present much of a challenge with regard to its contents. However, its characteristics show that several outbound messages are transformed into a target message. To see this, navigate to the SIGNATURE tab, where the two used messages can be found, including their occurrences. For the outbound message, set an occurrence of 0..UNBOUNDED, and then return to the DEFINITION tab (see Figure 4.61).

Now map the three fields to one another and connect the node Z_RFC_MATERIALINFO_## on the outbound side to the target node MatInfo, as shown in Figure 4.62. Save the mapping and test it by duplicating the subtree Z_RFM_MATERIALINFO_## on the outbound side, using the menu and populating it with data.



Source Message(s) *

Software Component Version: SC_TRAINING_PI_00 1 of www.sap-press.com

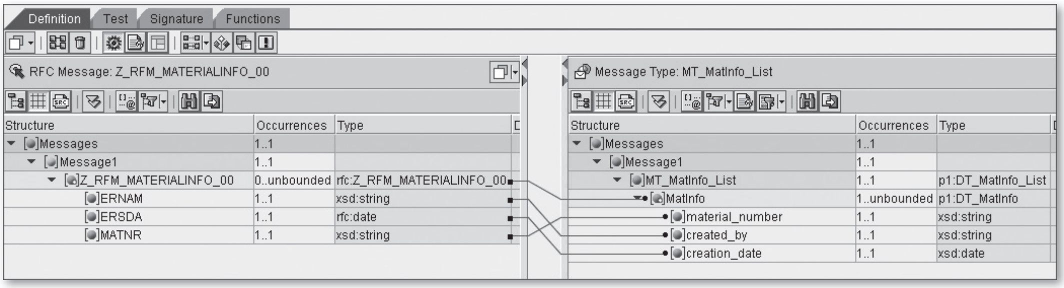
Type	Name	Namespace	Occurrence
RFC Message	Z_RFM_MATERIALINFO_00	urn:sap-com:	0..unbounded

Target Message(s) *

Software Component Version: SC_TRAINING_PI_00 1 of www.sap-press.com

Type	Name	Namespace	Occurrence
Message Type	MT_MatInfo_List	http://www.s	1

Figure 4.61 Setting the Occurrence in the Message Mapping



RFC Message: Z_RFM_MATERIALINFO_00

Structure	Occurrences	Type
Messages	1..1	
Message1	1..1	
Z_RFM_MATERIALINFO_00	0..unbounded	rfc:Z_RFM_MATERIALINFO_00
ERNAM	1..1	xsd:string
ERSDA	1..1	rfc:date
MATNR	1..1	xsd:string

Message Type: MT_MatInfo_List

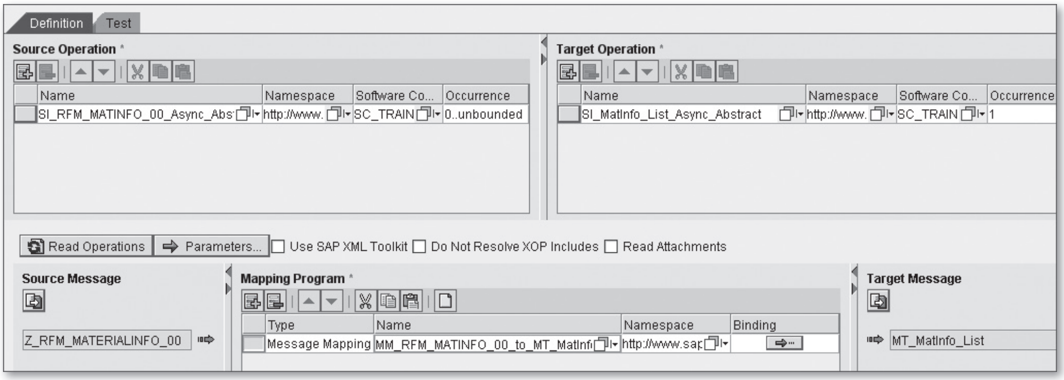
Structure	Occurrences	Type
Messages	1..1	
Message1	1..1	
MT_MatInfo_List	1..1	p1:DT_MatInfo_List
MatInfo	1..unbounded	p1:DT_MatInfo
material_number	1..1	xsd:string
created_by	1..1	xsd:string
creation_date	1..1	xsd:date

Figure 4.62 Message Mapping of the ccBPM Exercise

Creating the operation mapping

This message mapping still needs to be integrated in an appropriate operation mapping. The special thing about this mapping is that both the sending and the receiving interface belong to the Abstract type. Still, you can handle and map these interfaces as usual.

Create the operation mapping OM_RFM_MATINFO_##_Async_Abstract_to_MT_MatInfo_List_Async_Abstract and use SI_RFM_MATINFO_##_Async_Abstract as the outbound interface. The target interface is of the SI_MatInfo_List_Async_Abstract type. The occurrence setting you just specified for the message mapping also needs to be specified for the operation mapping. You can set the OCCURRENCE of the outbound interface directly in the DEFINITION tab to a value of 0...UNBOUNDED. In the lower area, import the interfaces and select the message mapping you just created (see Figure 4.63).



Source Operation *

Name	Namespace	Software Co...	Occurrence
SI_RFM_MATINFO_00_Async_Abs	http://www.	SC_TRAIN	0..unbounded

Target Operation *

Name	Namespace	Software Co...	Occurrence
SI_MatInfo_List_Async_Abstract	http://www.	SC_TRAIN	1

☐ Use SAP XML Toolkit ☐ Do Not Resolve XOP Includes ☐ Read Attachments

Source Message: Z_RFM_MATERIALINFO_00

Mapping Program *

Type	Name	Namespace	Binding
Message Mapping	MM_RFM_MATINFO_00_to_MT_MatInf	http://www.sap	

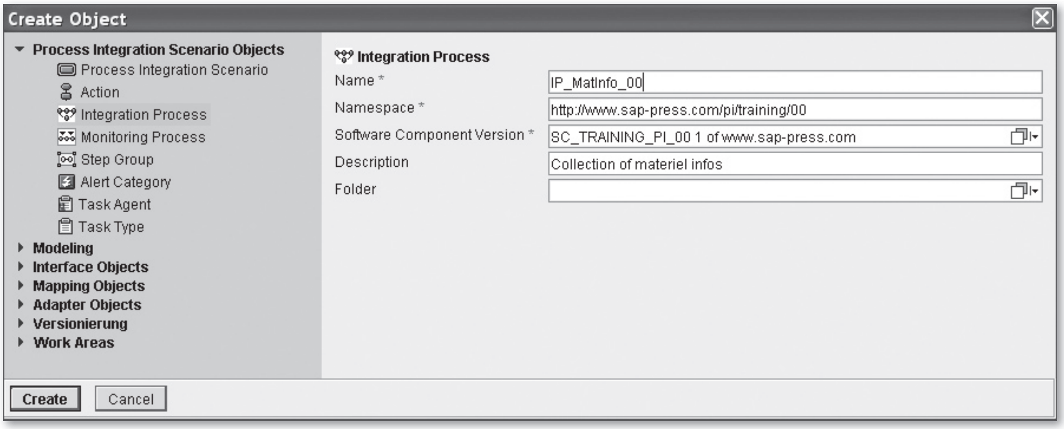
Target Message: MT_MatInfo_List

Figure 4.63 Setting the Occurrence in the Operation Mapping

Creating the Integration Process

Creating the integration process

You have now created all of the objects you will use during the integration process. Open the context menu of your namespace and choose the New option, or follow the OBJECT • NEW menu path, to get to the General Creation Wizard. In the wizard, navigate to the menu path: PROCESS INTEGRATION SCENARIO OBJECTS • INTEGRATION PROCESS. Make sure that your namespace and software component version are displayed. Name the new integration process `IP_MatInfo_##`, and enter a description (see Figure 4.64). For this object, it is necessary to assign the participant number, because your process will later be visible in the entire SAP NetWeaver PI.



Integration Process

Name *	IP_MatInfo_00
Namespace *	http://www.sap-press.com/pi/training/00
Software Component Version *	SC_TRAINING_PI_00 1 of www.sap-press.com
Description	Collection of materiel infos
Folder	

Figure 4.64 Creating a New Integration Process

creating the object brings you to the Details window, which is a graphical process editor. Detach the window using the Fixing pin icon, and hide the header to have as much space as possible (see Figure 4.65).

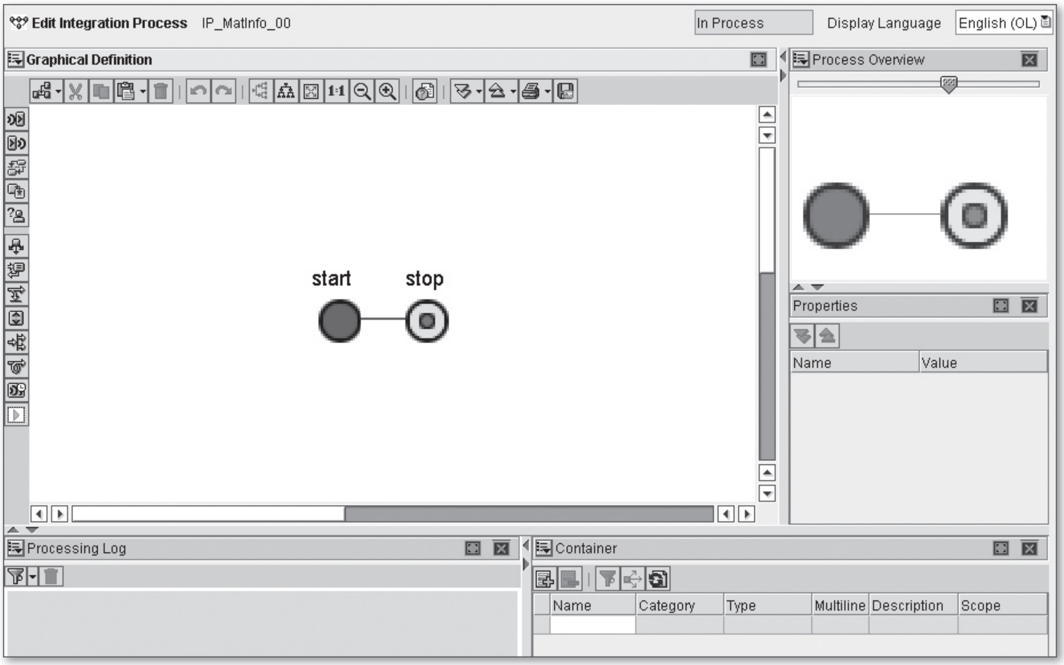


Figure 4.65 Structure of the Graphical Process Editor

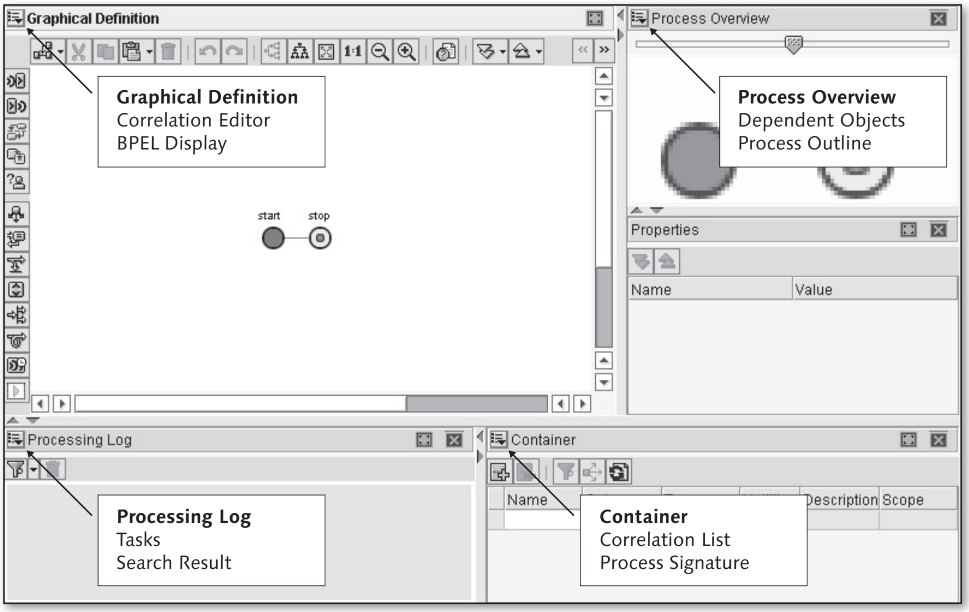
Structure of the process editor

The editor itself is divided into several areas. The most obvious element is the graphical work area, where you can add objects by dragging and dropping them from the list on the left side. You will find functions for controlling the view at the top of the work area.

The space to the right of the work area is divided into two parts: The upper area contains an overview for particularly large integration processes. (The slider lets you set the zoom.) Below the PROCESS OVERVIEW section, the PROPERTIES section shows the elements selected in the work area. In the bottom-left section of the editor is the PROCESSING LOG, which displays error messages and annotations. To the right of the log is a list of the CONTAINER elements. These containers are the process-internal representation of interfaces or variables you can create for internal use.

Alternative functions of the panes

All of the panes provide additional alternative functions, which you can select using the list icon next to the name of the pane. If you have already worked with SAP business workflows, the principles of the process editor will remind you of the workflow builder. In this exercise, you will only get to know the work area and container panes; you will create a correlation in another function. An overview of all of the possible views in the various panes is shown in Figure 4.66.



Graphical Definition
Correlation Editor
BPEL Display

Process Overview
Dependent Objects
Process Outline

Processing Log
Tasks
Search Result

Container
Correlation List
Process Signature

Figure 4.66 Views of the Panes in the Process Editor

Creating the container elements

Let's start by creating the container elements. As mentioned earlier, the container elements contain objects that are only used within the process. In the case of messages, however, these internal objects equal an abstract interface. A particular aspect of these message container elements is that they can also exist in a multiline format; this means that a list of messages can be addressed with a single element. However, these multiline containers can only exist within the process; they can't be sent in their original format.

Scheme of the message processing

For this integration process, you will need three message container elements. The first container element, `MatInfo`, receives the incoming messages with the material information. This element is instantiated up to three times over the course of the integration process. The individual material information will be appended after receipt to the second element, `MatInfoContainer`. It functions as a collecting list of material information because of its multiline property. However, this list can only be used within the integration process.

A conversion into the third message element, `MatInfo_List`, has to take place to allow the data to leave the process. This element is based on a service interface that can contain information about multiple materials, and thus does not require the multiline property in the integration process. The messages, and the containers used as their interrelationships, are shown schematically in Figure 4.67.

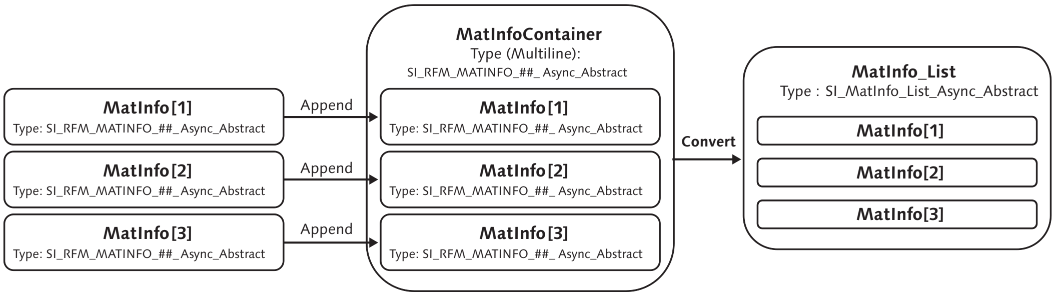

MatInfoContainer
Type (Multiline):
SI_RFM_MATINFO_##_ Async_Abstract
MatInfo[1]
Type: SI_RFM_MATINFO_##_Async_Abstract
MatInfo[2]
Type: SI_RFM_MATINFO_##_Async_Abstract
MatInfo[3]
Type: SI_RFM_MATINFO_##_Async_Abstract
Append
Append
Append
MatInfo[1]
Type: SI_RFM_MATINFO_##_ Async_Abstract
MatInfo[2]
Type: SI_RFM_MATINFO_##_ Async_Abstract
MatInfo[3]
Type: SI_RFM_MATINFO_##_ Async_Abstract
Convert
MatInfo_List
Type : SI_MatInfo_List_Async_Abstract
MatInfo[1]
MatInfo[2]
MatInfo[3]

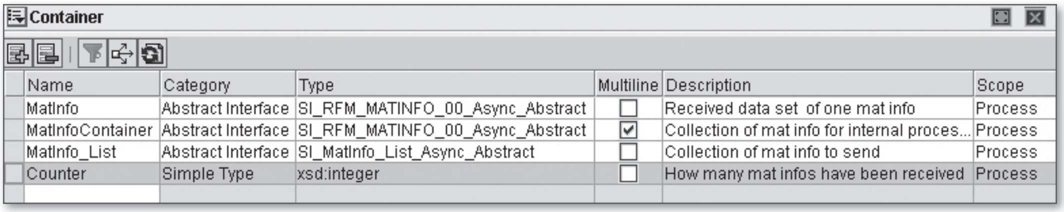
Figure 4.67 Use of the Container Elements in the Integrations Process

The first container, called `MatInfo`, belongs to the ABSTRACT INTERFACE category and uses the interface `SI_RFM_MATINFO_##_Async_Abstract`. You can find this entry in the input help of the TYPE column, which only lists abstract interfaces of your software component version. This container always contains the message currently sent to the process by the RFC sender.

Note that the MULTILINE column can be used to make this container a multiline container (this is not used for the first container object). The DESCRIPTION column allows you to give the object a description to reflect its function. The SCOPE column displays the validity area of the element. Usually, the entire process is the validity area. You can only restrict the validity of these blocks if you insert the blocks in the work area.

Using the same pattern, create the `MatInfoContainer` element, which belongs to the same category and type, but is a multiline element. This element stores `MatInfo` messages until they are sent in a single message. Before being sent, however, this element must be converted, due to its multiline character. The target element of this conversion is `MatInfo_List`, which also belongs to the ABSTRACT INTERFACE category, but references the `SI_MatInfo_List_Async_Abstract` type.

Last but not least, you need a variable that counts how many times the loop runs. This variable should be named COUNTER, and should be of the simple `xsd:integer` type. The list of your container elements should now correspond to the one shown in Figure 4.68.



Name	Category	Type	Multiline	Description	Scope
MatInfo	Abstract Interface	SI_RFM_MATINFO_00_Async_Abstract	☐	Received data set of one mat info	Process
MatInfoContainer	Abstract Interface	SI_RFM_MATINFO_00_Async_Abstract	✓	Collection of mat info for internal proces...	Process
MatInfo_List	Abstract Interface	SI_MatInfo_List_Async_Abstract	☐	Collection of mat info to send	Process
Counter	Simple Type	xsd:integer	☐	How many mat infos have been received	Process

Figure 4.68 Container Elements of the Integration Process

Creating the correlation

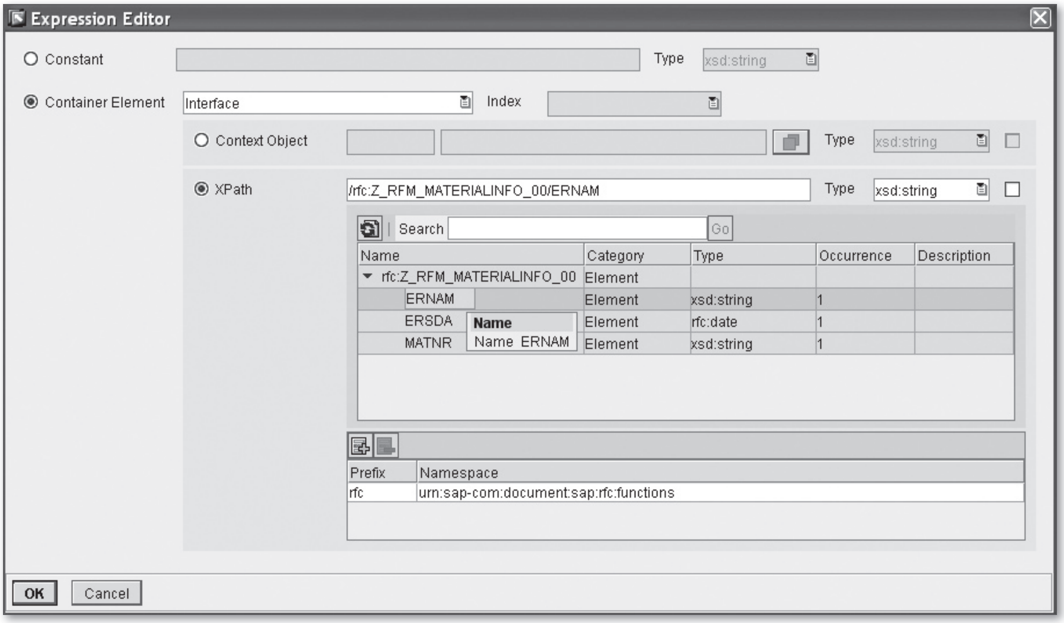
Before you get to the graphical modeling, you need to create a correlation. Using this correlation, the process can identify related messages by their content. In this exercise, the process must map and collect messages from the same creator; thus, the correlation consists of the `created_by` field contents of the incoming messages.

To create a correlation, click the SWITCH EDITOR icon, and select the alternative function Correlation List of the CONTAINER pane. Enter the name `MatInfo` for the correlation. Highlight the new entry and click on the

DETAILS icon above the list. The graphical work area changes into the Correlation Editor view.

In the upper area of this view, the newly created correlation is already selected. The area below it is divided into three parts: On the left, you can see the correlation containers of the correlation fields. The center part lets you select involved messages. The right area lets you specify the actual fields.

In the left area, specify the name `created_by` and maintain the `xsd:string` type. The message to which the correlation should apply is the `SI_RFM_MATINFO_##_Async_Abstract` type, which can be specified in the center area. By selecting the interface, it is also displayed in the right area, which up to now wasn't available for input. Open the input help next to the Created_by field, thus opening the EXPRESSION EDITOR. Select the `ERNAM` element from the displayed message. For this, you must first activate the CONTAINER ELEMENT option and make sure that the INTERFACE entry is shown in the selection menu (see Figure 4.69).

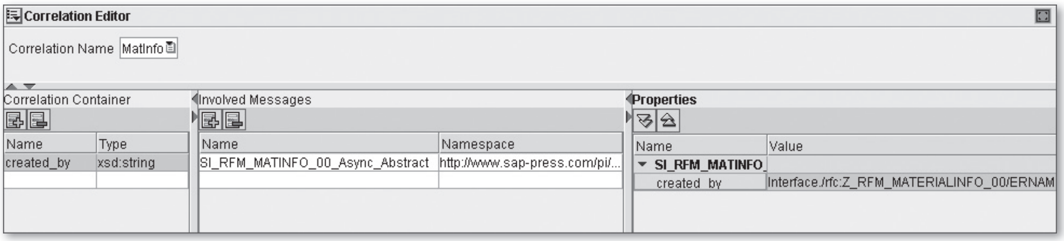


Name	Category	Type	Occurrence	Description
rfc:Z_RFM_MATERIALINFO_00	Element			
ERNAM	Element	xsd:string	1	
ERSDA	Element	rfc:date	1	
MATNR	Element	xsd:string	1	

Prefix	Namespace
rfc	urn:sap-com:document:sap:rfc:functions

Figure 4.69 Expression Editor in the Correlation Editor

Accept the selection by clicking the OK button. You are then brought back to the correlation editor, as shown in Figure 4.70.



Name	Type
created_by	xsd:string

Name	Namespace
SI_RFM_MATINFO_00_Async_Abstract	http://www.sap-press.com/pi/...

Name	Value
SI_RFM_MATINFO_	
created_by	Interface./rfc:Z_RFM_MATERIALINFO_00/ERNAM

Figure 4.70 Correlation Editor

Click the SWITCH EDITOR icon on the top left of the CORRELATION EDITOR to change back to the Graphical Definition view. You can now begin with the graphical modeling of the business process. Table 4.11 gives you an overview of all of the available objects (or step types) in the process editor. However, you will only use a few of them in this exercise.

Icon	Step Type
	Receive: This step lets you receive messages of a particular interface, and always represents the beginning of an integration process. The reception always happens asynchronously, but it can simulate synchronous communication using a synchronous-asynchronous bridge (Sync/Async Bridge). To bring messages together, correlations can be specified.
	Send: This step sends a message from a specific interface in a synchronous or asynchronous way. Send steps can close a Sync/Async Bridge. Correlations can be specified to bring messages together.
	Receiver Determination: The recipient identification step is used to get a list of recipients of a particular message. The configured receiver determination from the Integration Directory is used.
	Transformation: A transformation can be used to convert messages of one interface into messages of another interface. It can both split and merge several messages.

Table 4.11 Step Types in the Process Editor

Icon	Step Type
	User Decision: The user decision step is new to SAP NetWeaver PI, but has been used in SAP Business Workflows for a long time. During runtime, agents are determined based on a configurable parameter. An agent can choose from preset options that affect the further course of the integration process. The decision template can include the values of other container objects.
	Switch: A switch can be used to distinguish several values of a container element.
	Container Operation: A container operation is used to change container objects. Two different operations can be distinguished: the attachments of an object to another, multiline object; and the change in the value of a container object (such as a counter variable).
	Control: The control step can stop the integration process, raise an exception, or send an alert message.
	Block: A block can be used for logical fragmentation of integration processes, and is partly implicitly inserted by other steps. A block can also be used for reducing the visibility of container objects.
	Fork: A parallel section allows independent processing of multistep sequences. Each step sequence can therefore be linked to a certain condition that must be met to perform it. After processing all of the steps of the different sequences, they reunite in a union operator.
	Loop: In an integration process, the loop works like a WHILE loop. As long as the condition specified in this step is not met, the steps within the loop are processed one after another.
	Wait: A wait step can delay the execution of subsequent steps. You can specify a fixed date or a time period.
	Undefined: An undefined step serves as a placeholder, or for testing purposes.

Table 4.11 Step Types in the Process Editor (Cont.)

Creating the receive loop

The first step in a business process must always be a Receive step, which can either be situated at the very beginning, or be integrated in a loop. Because this process receives exactly three messages, the most obvious procedure is to use a loop.

Drag a loop from the left toolbar to the work area between the START and STOP objects. The place where you can drop the loop is indicated by yellow parentheses. The new loop element is automatically highlighted, and its properties are displayed to the right of the work area (see Figure 4.71). Name this object COLLECTION OF MATERIAL INFORMATION. The CONDITION field specifies how long this loop is supposed to run, therefore making it a `WHILE` loop.

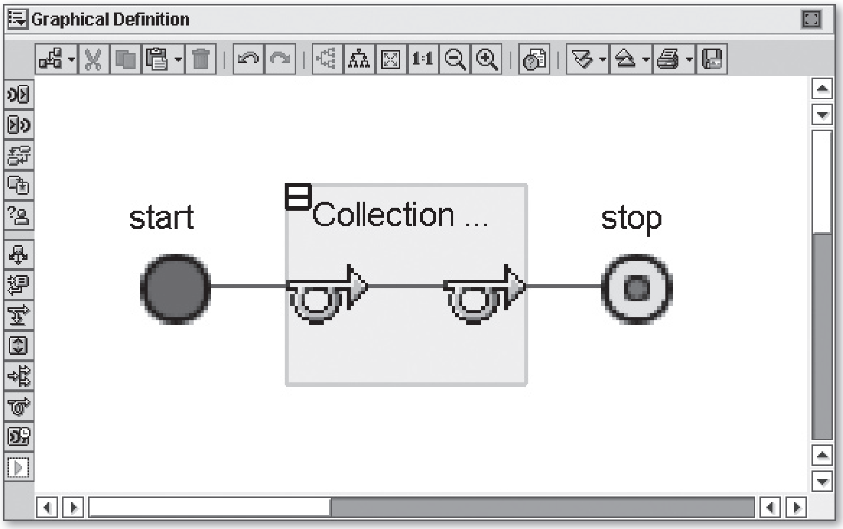
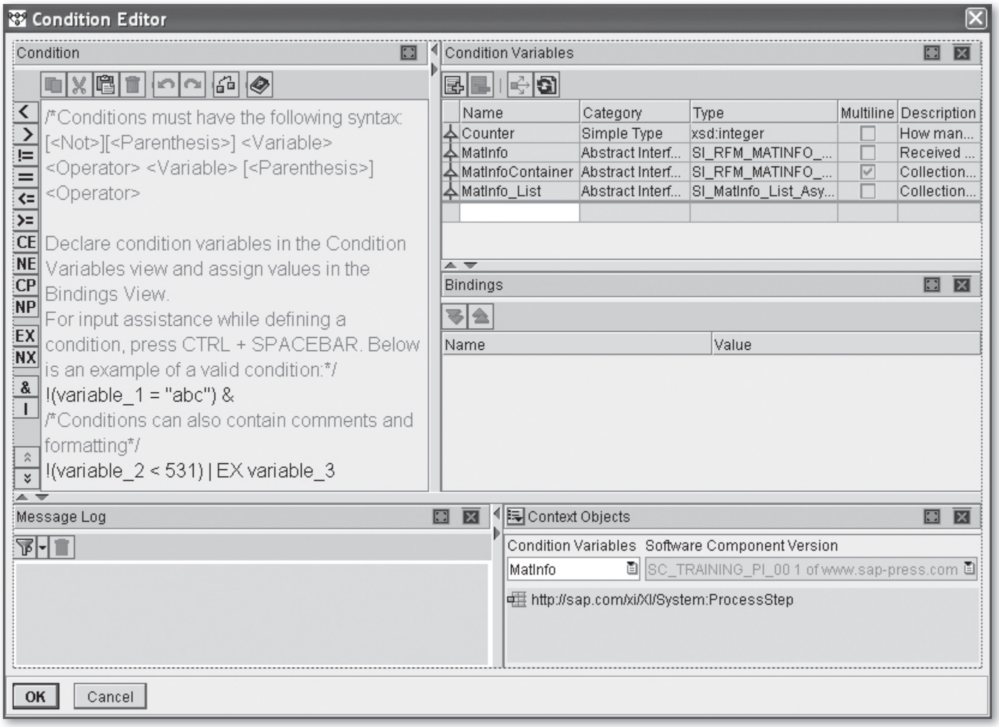


Figure 4.71 Integration Process after the Insertion of the Loop

Specification of the exit condiition for the loop

Click on the white area and open the input help; the condition editor is displayed. Set the loop so it runs, as long as the COUNTER variable is not 3. In contrast to SAP XI, the condition editor in SAP NetWeaver PI has changed significantly. On one hand, the intuitive creation of simple terms is more complicated; on the other, the power of the editor has increased due to new flexibility. If a CONDITION field has no content, a help text is displayed in the actual working area of the editor, which will facilitate the first steps (see Figure 4.72). Once you click the text box to define the condition, the text disappears, and the field accepts input.



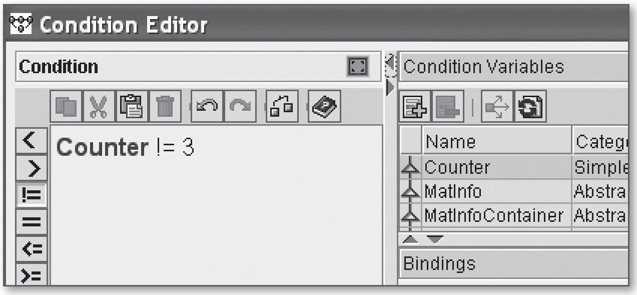
```
/*Conditions must have the following syntax:
[<Not>][<Parenthesis>] <Variable>
<Operator> <Variable> [<Parenthesis>]
<Operator>

Declare condition variables in the Condition
Variables view and assign values in the
Bindings View.
For input assistance while defining a
condition, press CTRL + SPACEBAR. Below
is an example of a valid condition:*/
!(variable_1 = "abc") &
/*Conditions can also contain comments and
formatting*/
!(variable_2 < 531) | EX variable_3
```

Name	Category	Type	Multiline	Description
Counter	Simple Type	xsd:integer	☐	How man...
MatInfo	Abstract Interf...	SI_RFM_MATINFO_...	☐	Received ...
MatInfoContainer	Abstract Interf...	SI_RFM_MATINFO_...	✓	Collection...
MatInfo_List	Abstract Interf...	SI_MatInfo_List_Asy...	☐	Collection...

Condition Variables: MatInfo
Software Component Version: SC_TRAINING_PI_00 1 of www.sap-press.com
http://sap.com/xi/XI/System:ProcessStep

Figure 4.72 Condition Editor of the Loop Step in Initial State

In the CONDITION VARIABLES area, choose the COUNTER container object, and drag it into the left CONDITION field. The explanatory text disappears, and instead displays the name of the inserted container object. Then drag the inequality button (third from the top) on the left edge into the CONDITION field, and drop it behind the container element. Finally, enter the constant value of 3, so the condition `Counter! = 3` can be seen in the condition field (see Figure 4.73).



Counter != 3

Name	Categ
Counter	Simple
MatInfo	Abstra
MatInfoContainer	Abstra

Figure 4.73 Condition in the Receiving Loop

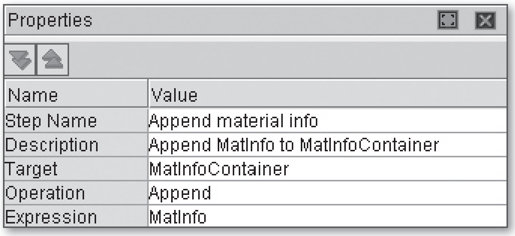
For better readability, you should separate the operands and operator by a space. Perform a semantic test using the appropriate icon above the condition field; if an error is found, an error message will be displayed on the left-bottom. Copy the condition by clicking the OK button; the condition should now be displayed in the appropriate field in the properties of the loop step.

Inserting the steps in the loop

Now insert a Receive step in the loop, and call it Receive material info. Because this is the first Receive step in the process, the Start Process property is automatically set for this object. In the properties, set the `MatInfo` container element as the Message. The Receive step then expects a message of this type. Specify the `MatInfo` correlation as the used and activated correlation; incoming messages are then grouped using this correlation, depending on the creator of the material. This means that for messages with the same creator, a specific instance of the loop is only called after the first run.

Appending a incoming message to the multiline container

Now let's append an incoming message to the `MatInfoContainer` container element, which is accomplished by adding a Container Operation step. This step allows you to make changes to containers, such as setting variables or appending messages to lists. Add the new object after the Receive step and name it APPEND MATERIAL INFO. The target of this operation is the `MatInfoContainer` container element, to which the `MatInfo` container element is appended. Note that the operation must be APPEND, not Assign (see Figure 4.74).



Properties

Name	Value
Step Name	Append material info
Description	Append MatInfo to MatInfoContainer
Target	MatInfoContainer
Operation	Append
Expression	MatInfo

Figure 4.74 Properties of the Container Operation for Appending MatInfo to MatInfoContainer

The last object in the loop is another Container Operation, which increases the variable `counter` by one; name the Container Operation, Increase Counter. The target of this operation is the `Counter` container element. This time, the Assign operation is appropriate. As the first expression, reselect the `Counter` container element. The operator for the addition is a plus sign (+). The second expression is a constant of the type `xsd:integer` with a value of 1. The property values of this object result in the mathematical expression `Counter = Counter +1`. After completing the loop, the integration process should have the step sequence shown in Figure 4.75.

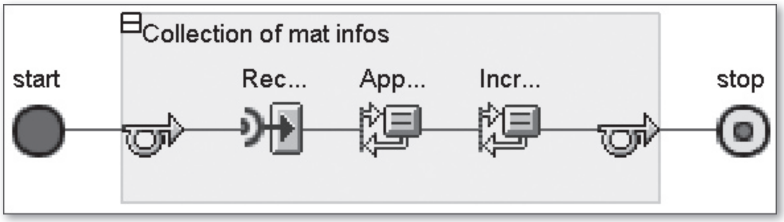


Figure 4.75 Step Sequence of the Integration Process after Completing the Loop

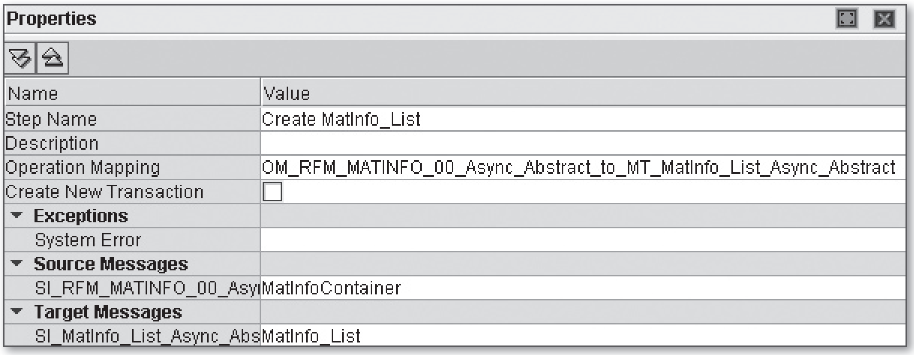
Conversion and delivery of the outbound message

The two missing steps merge the multiline container element into a single message before sending it. Before the loop, insert a Transformation object and call it `Create MatInfo_List`. Transformation objects allow you to convert messages to a different format. For this, specify the operation mapping `OM_RFM_MATINFO_##_Async_Abstract_to_MT_MatInfo_List_Async_Abstract`. After specifying the operation mapping, appropriate fields for the properties are completed, and the expected message type is shown. At process execution time, the source messages reside in the

`MatInfoContainer` container element. In this transformation, the target message is passed to the `MatInfo_List` element (see Figure 4.76).

As the name suggests, the last step, Send Target Message, is a Send step. The container element `MatInfo_List` is sent asynchronously.

The entire structure of the integration process is illustrated in Figure 4.77. Save the process and verify it via the menu path: INTEGRATION PROCESS • CHECK. If the process is successful, you should get a note that `MatInfo` is initialized, but not used. If there are no more errors, you can activate all new objects.



Properties

Name	Value
Step Name	Create MatInfo_List
Description	
Operation Mapping	OM_RFM_MATINFO_00_Async_Abstract_to_MT_MatInfo_List_Async_Abstract
Create New Transaction	☐
Exceptions	
System Error	
Source Messages	
SI_RFM_MATINFO_00_Asy	MatInfoContainer
Target Messages	
SI_MatInfo_List_Async_Abs	MatInfo_List

Figure 4.76 Properties of the Transformation Step for MatInfo_List

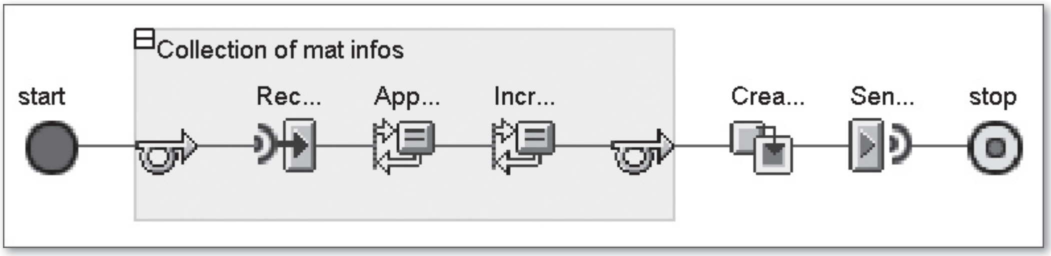


Figure 4.77 Structure of the Complete Integration Process

4.4.3 Configuration

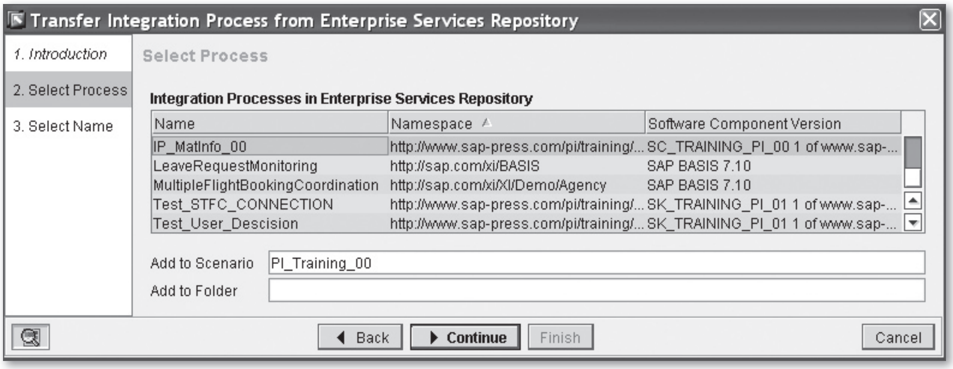
Overview of the configuration scenarios

Message processing is configured in two steps, because an integration process is equal to a business system with regard to configuration. The RFC call must be forwarded to the process; for this, use the RFC sender channel you created in Exercise 1. (Because they run in SAP NetWeaver

PI, integration processes don't need adapters.) You also need to configure the sending of outbound messages to system B, so you will need to create two configuration scenarios.

Creating the integration process in the Integration Directory

Before you can use the integration process during the configuration phase, you must declare it in the Integration Directory. In the Integration Directory, follow the menu path: COMMUNICATION COMPONENT • INTEGRATION PROCESS. Create a new process that represents your `IP_MatInfo_##` object from the Enterprise Services Repository. In the second step, select your configuration scenario, `PI_Training_##`, to assign the integration process to the corresponding scenario list (see Figure 4.78).


Transfer Integration Process from Enterprise Services Repository

1. Introduction
2. Select Process
3. Select Name

Select Process

Integration Processes in Enterprise Services Repository

Name	Namespace	Software Component Version
IP_MatInfo_00	http://www.sap-press.com/pi/training/...	SC_TRAINING_PI_00 1 of www.sap-...
LeaveRequestMonitoring	http://sap.com/xi/BASIS	SAP BASIS 7.10
MultipleFlightBookingCoordination	http://sap.com/xi/XI/Demo/Agency	SAP BASIS 7.10
Test_STFC_CONNECTION	http://www.sap-press.com/pi/training/...	SK_TRAINING_PI_01 1 of www.sap-...
Test_User_Descision	http://www.sap-press.com/pi/training/...	SK_TRAINING_PI_01 1 of www.sap-...

Add to Scenario: PI_Training_00
Add to Folder:

Figure 4.78 Selecting the Integrations Process and Assigning it to the Configuration Scenario

In the third step of the Definition Wizard, you can specify a different name for your process. To keep things simple, use the same name you used in the Enterprise Services Repository, `IP_MatInfo_##`. After completing the wizard, your process shows up in the menu path COMMUNICATION COMPONENT • INTEGRATION PROCESS.

Delivery of Inbound Messages to the Integration Process

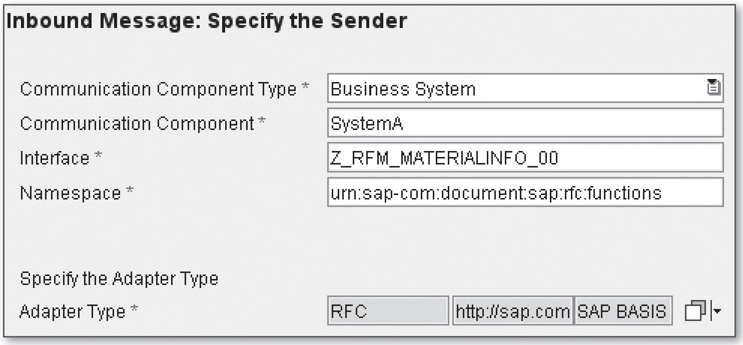
Table 4.12 provides an overview of all of the configuration objects used for delivering the creation notification to the integration process. Communication channels identified by an asterisk (*) were created in Section 4.1.3, and will be reused here.

Object Type	Sender Side: System A	Receiver Side: Integration process IP_MatInfo_##
Communication channel	RFC_Sender_Channel_## *	
Sender agreement	| SystemA | Z_RFM_MATERIALINFO_##|*|*	
Receiver determination	| SystemA | Z_RFM_MATERIALINFO_##	
Interface determination	|SystemA|Z_RFM_MATERIALINFO_##||IP_MatInfo_##	

Table 4.12 Elements in the Integration Directory for the First Scenario in the BPM Exercise

Calling the Configuration Wizard for the first scenario

Call the Configuration Wizard and select the Internal Communication option. The sender is business system A, which sends data using the RFC adapter with the RFC interface, Z_RFM_MATERIALINFO_## (see Figure 4.79).


Inbound Message: Specify the Sender

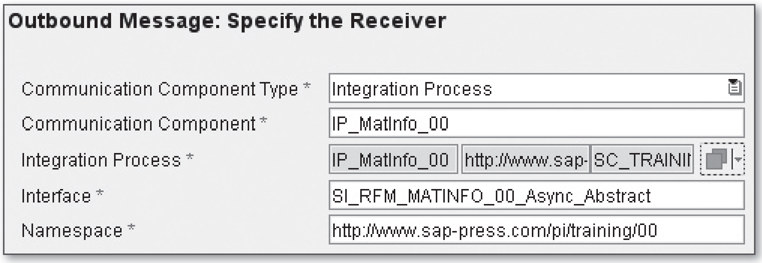
Communication Component Type *	Business System
Communication Component *	SystemA
Interface *	Z_RFM_MATERIALINFO_00
Namespace *	urn:sap-com:document:sap:rfc:functions

Specify the Adapter Type

Adapter Type *	RFC	http://sap.com	SAP BASIS

Figure 4.79 Sender Settings in the Configuration Wizard for the Incoming Message

The receiver is the IP_MatInfo_## integration process, which is addressed with the XI adapter. Once the integration process was selected as the COMMUNICATION COMPONENT using the value help, the associated integration process from the Enterprise Services Repository appears underneath. When you open the input help of the interface, only the SI_RFM_MATINFO_##_Async_Abstract interface is displayed; this is used by the Receive step to start the process. If the value help does not display any entries, you can directly enter the interface name and the namespace (see Figure 4.80).



Outbound Message: Specify the Receiver

Communication Component Type *	Integration Process
Communication Component *	IP_MatInfo_00
Integration Process *	IP_MatInfo_00 | http://www.sap- | SC_TRAINI
Interface *	SI_RFM_MATINFO_00_Async_Abstract
Namespace *	http://www.sap-press.com/pi/training/00

Figure 4.80 Receiver Settings in the Configuration Wizard for the Incoming Message

The sender agreement `|SystemA|Z_RFM_MATERIALINFO_##|*|*` uses the existing communication channel, `RFC_Sender_Channel_##`, of Exercise 1. The screen for creating the receiver determination, `|SystemA|Z_RFM_MATERIALINFO_##`, is of a purely informative nature.

In the next step, the interface determination `|SystemA|Z_RFM_MATERIALINFO_##||IP_MatInfo_##` displays the interface `SI_RFM_MATINFO_##_Async_Abstract` as the target, but does not find an operation mapping; this is not necessary because both the sending and receiving interface use the same message type. Because the receiver is integrated with the XI adapter, a receiver agreement is not required.

Assign all objects to your `PI_Training_##` scenario and exit the wizard. You can test the integration process after it has been activated, but the outbound message will not be delivered. Before testing the project, wait until the delivery of the outbound message has been configured.

Delivery of the Outbound Message of the Integration Process

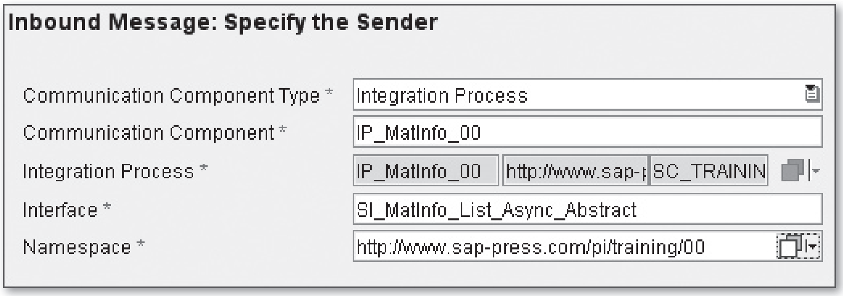
Delivery of the outbound message can be arranged in a way similar to the first configuration scenario of this exercise. The existing communication channel `File_ReceiverChannel_##`, which was created in Exercise 1, is used for the receiver. Table 4.13 provides an overview of all configuration objects used in this scenario.

Object Type	Sender Side: Integration Process IP_MatInfo_##	Receiver Side: System B
Communication channel		`File_ReceiverChannel_##*`
Receiver agreement	`|IP_MatInfo_##||SystemB|SI_MatInfo_List_Async_In`	
Receiver determination	`|IP_MatInfo_##|SI_MatInfo_List_Async_Abstract`	
Interface determination	`|IP_MatInfo_##|SI_MatInfo_List_Async_Abstract||SystemB`	

Table 4.13 Elements in the Integration Directory for the Outbound Message of the Integration Process

Calling the Configuration Wizard for the second scenario

Call the Configuration Wizard once more and select Internal Communication. The sender of this scenario is the integration process `IP_MatInfo_##`. Help is again restricted, because only the used interfaces are displayed when sending messages from the process. In this case, only one message is sent, so only one interface is displayed. Select the displayed interface `SI_MatInfo_List_Async_Abstract` (see Figure 4.81).



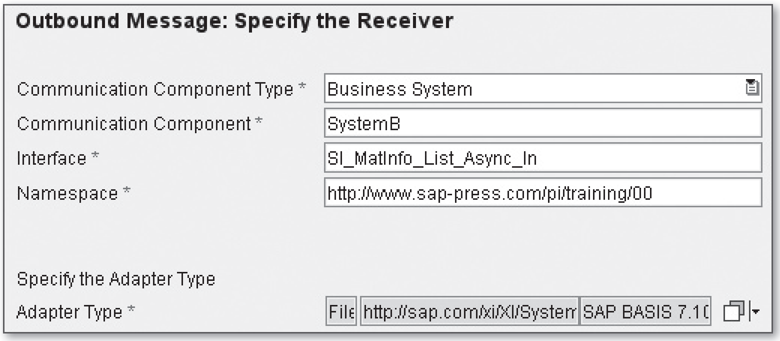
Inbound Message: Specify the Sender

Communication Component Type *	Integration Process
Communication Component *	IP_MatInfo_00
Integration Process *	IP_MatInfo_00 | http://www.sap-p | SC_TRAININ
Interface *	SI_MatInfo_List_Async_Abstract
Namespace *	http://www.sap-press.com/pi/training/00

Figure 4.81 Sender Settings in the Configuration Wizard of the Outbound Message

Specification of the receiver in the Configuration Wizard

The outbound message receiver is a File adapter of business system B, which responds to the `SI_MatInfo_List_Async_In` interface (see Figure 4.82). This interface is the direction-related counterpart to the abstract interface of the outbound message. A sender agreement isn't used, because the sending process runs in SAP NetWeaver PI.


Outbound Message: Specify the Receiver

Communication Component Type *	Business System
Communication Component *	SystemB
Interface *	SI_MatInfo_List_Async_In
Namespace *	http://www.sap-press.com/pi/training/00

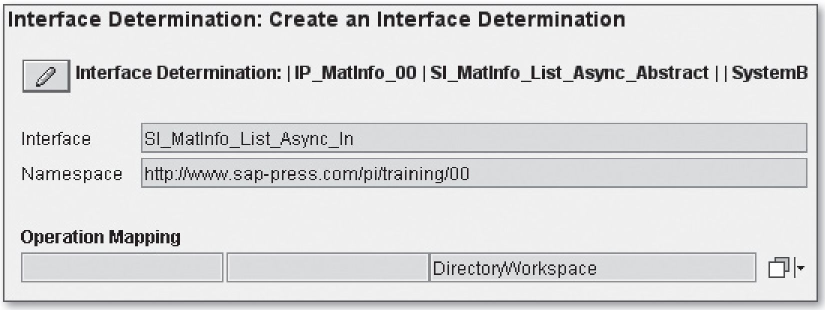
Specify the Adapter Type

Adapter Type * File | http://sap.com/xi/XI/System | SAP BASIS 7.10

Figure 4.82 Receiver Settings in the Configuration Wizard for the Outbound Message

Information about the receiver determination

The receiver determination step informs you that system B is integrated in the object |IP_MatInfo_##|SI_MatInfo_List_Async_Abstract. As with the first scenario, the interface determination |IP_MatInfo_##|SI_MatInfo_List_Async_Abstract||SystemB doesn't require an operation mapping, because it uses the same message type (see Figure 4.83).


Interface Determination: Create an Interface Determination

Interface Determination: | IP_MatInfo_00 | SI_MatInfo_List_Async_Abstract | | SystemB

Interface	SI_MatInfo_List_Async_In
Namespace	http://www.sap-press.com/pi/training/00

Operation Mapping

DirectoryWorkspace

Figure 4.83 Interface Determination of the Outbound Message

The receiver agreement |IP_MatInfo_##||SystemB|SI_MatInfo_List_Async_In uses the existing communication channel File_ReceiverChannel_##.

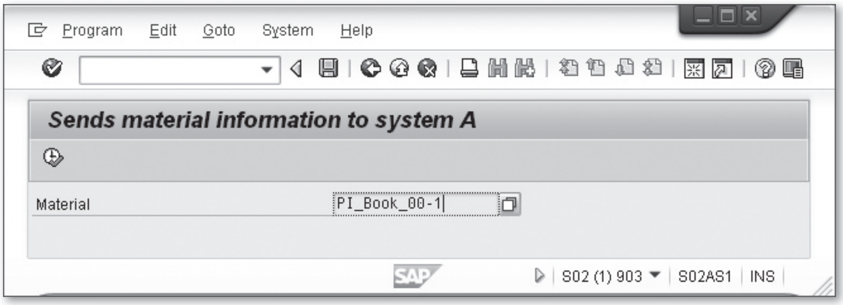
Add all of the new objects to your scenario PI_Training_## and finish the wizard. You can now activate all new configuration objects. Now that you've finished the fourth exercise, test the process.

4.4.4 Process and Monitoring

The process of this integration scenario starts in system A. As such, log in as user SYS_A-## and call Transaction SA38.

Sending the first material creation notification

Run the program Z_PROG_MATERIALINFO_##. The program calls the function module Z_RFM_MATERIALINFO_## in the background, using the RFC destination SYSTEMA_SENDER-##.[8] Within the program you can only specify a material number. Information about the creator, including the creation date, is sent along automatically. The material you specify isn't important at this point, because the correlation of the three messages is done using the creator (i.e., your SAP user SYS_A-##). Simply enter a name and run the program (see Figure 4.84).



Sends material information to system A

Material PI_Book_00-1

S02 (1) 903 | S02AS1 | INS

Figure 4.84 Call of Program Z_PROG_MATERIALINFO_##

You will see a message notifying you that the function module was successfully called, but this does not imply that the message was actually received; it only informs you that function module Z_RFM_MATERIALINFO_## has been called without any errors.

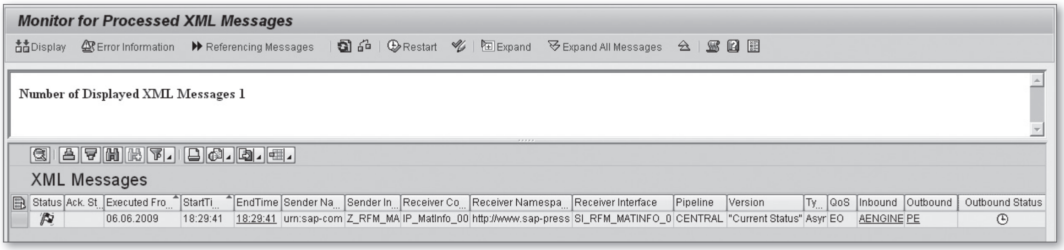
Checking the processing of the first message

Now change to the PI system and call Transaction SXMB_MONI. You can see that your message came in and has been successfully processed. If you scroll the display a bit to the right, you can see that the OUTBOUND STATUS column displays a clock symbol (see Figure 4.85).[9] This means that the message is currently being sent. This display remains unchanged

8 The direct call of the function module in Transaction SE37 starts a synchronous message exchange, and can thus cause corresponding errors.

9 When calling the transaction in English, the heading of this column may be "c." This depends on the support package being used.

until three messages from the same creator have come in, and the outbound message can be created.



Number of Displayed XML Messages 1

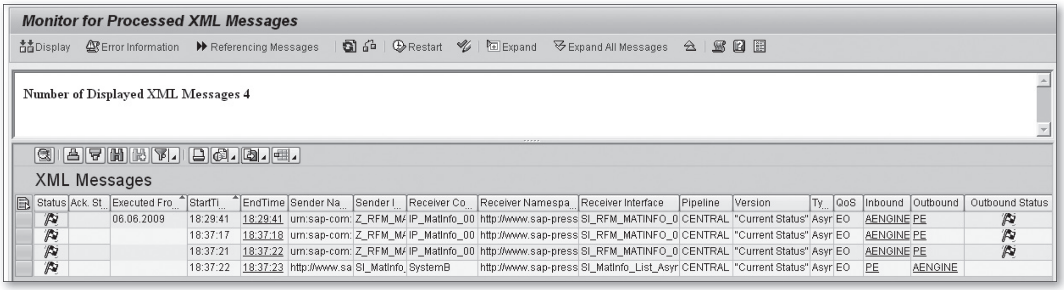
XML Messages

Status	Ack. St...	Executed Fro...	StartTi...	EndTime	Sender Na...	Sender In...	Receiver Co...	Receiver Namespa...	Receiver Interface	Pipeline	Version	Ty...	QoS	Inbound	Outbound	Outbound Status
🏁		06.06.2009	18:29:41	18:29:41	urn:sap-com	Z_RFM_MA	IP_MatInfo_00	http://www.sap-press	SI_RFM_MATINFO_0	CENTRAL	"Current Status"	Asyr	EO	AENGINE	PE	🕒

Figure 4.85 Message Monitoring after the First Message

Sending the two remaining messages

Call the program for notifying the material manager twice more, and return to the monitoring. You should now see the three incoming notifications, and a fourth message. The outbound status of the first three messages now displays a black-and-white checkered flag (see Figure 4.86), which lets you know that the process has completed. In the PI system, call Transaction AL11, Transaction ZAPCMD,[10] or the file tool provided on the website of this book to see if a file containing the corresponding message has been created.



Number of Displayed XML Messages 4

XML Messages

Status	Ack. St...	Executed Fro...	StartTi...	EndTime	Sender Na...	Sender I...	Receiver Co...	Receiver Namespa...	Receiver Interface	Pipeline	Version	Ty...	QoS	Inbound	Outbound	Outbound Status
🏁		06.06.2009	18:29:41	18:29:41	urn:sap-com:	Z_RFM_MA	IP_MatInfo_00	http://www.sap-press	SI_RFM_MATINFO_0	CENTRAL	"Current Status"	Asyr	EO	AENGINE	PE	🏁
🏁			18:37:17	18:37:18	urn:sap-com:	Z_RFM_MA	IP_MatInfo_00	http://www.sap-press	SI_RFM_MATINFO_0	CENTRAL	"Current Status"	Asyr	EO	AENGINE	PE	🏁
🏁			18:37:21	18:37:22	urn:sap-com:	Z_RFM_MA	IP_MatInfo_00	http://www.sap-press	SI_RFM_MATINFO_0	CENTRAL	"Current Status"	Asyr	EO	AENGINE	PE	🏁
🏁			18:37:22	18:37:23	http://www.sa	SI_MatInfo_	SystemB	http://www.sap-press	SI_MatInfo_List_Asyr	CENTRAL	"Current Status"	Asyr	EO	PE	AENGINE	

Figure 4.86 Message Monitoring after Three Messages of the Same Creator

Monitoring the business process

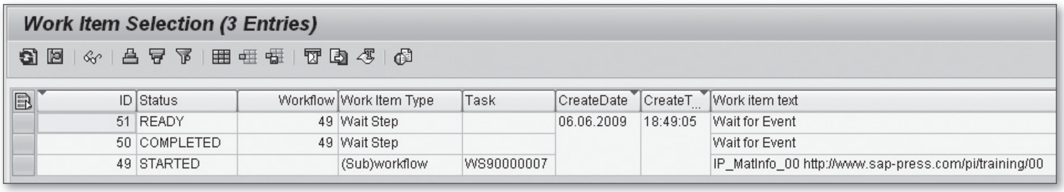
If the fourth message is not displayed in the monitoring, look at the monitoring of the business processes. In the XI system, call Transaction SXMB_MONI_BPE and double-click on the Process Selection entry. In the selection mask of the Service field, you can enter integration process

10 Detailed information on the SAP Commander (Transaction ZAPCMD) can be found at: *http://code.google.com/p/sapcommander*.

IP_MatInfo_## to only see the status of the workflow. You should not do this if you want to examine an incomplete or possibly erroneous workflow; instead, try to limit the selection period to only the time when the messages have been sent, and run the query. The work items processed during the selected period are now displayed.

Analyzing the work items

Figure 4.87 shows the status of work items and workflow after sending the first message. A Wait step of the loop has been processed by receiving the first message, and is marked as COMPLETED. Another Wait step awaits further messages. In addition, you can see that the workflow is started.

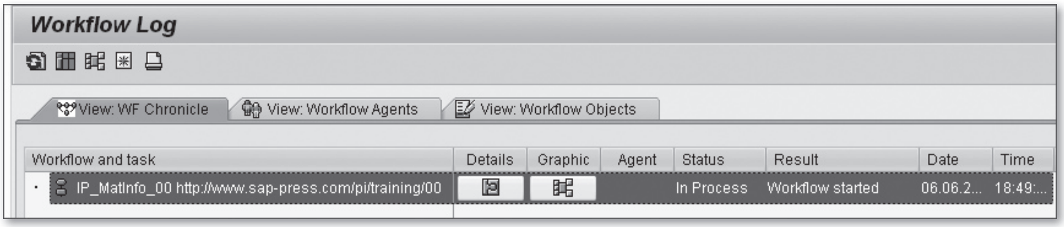


Work Item Selection (3 Entries)

ID	Status	Workflow	Work Item Type	Task	CreateDate	CreateT...	Work item text
51	READY	49	Wait Step		06.06.2009	18:49:05	Wait for Event
50	COMPLETED	49	Wait Step				Wait for Event
49	STARTED		(Sub)workflow	WS90000007			IP_MatInfo_00 http://www.sap-press.com/pi/training/00

Figure 4.87 Work Items after Sending a Message

To get to the workflow protocol, you can either select one of the steps by double-clicking it and then clicking on the LOG icon, or simply double-click the workflow entry in the collection. Regardless of which way you proceed, it brings you to the workflow protocol (see Figure 4.88).



Workflow Log

Workflow and task	Details	Graphic	Agent	Status	Result	Date	Time
IP_MatInfo_00 http://www.sap-press.com/pi/training/00				In Process	Workflow started	06.06.2...	18:49:...

Figure 4.88 Display of the Workflow Protocols

Graphical display of the workflow processing

Information on the current status of the workflow is displayed by clicking the button in the DETAILS column in the lower part of the screen. Click on the button in the GRAPHIC column. This brings you to the graphical display of the business process (see Figure 4.89). The green arrows allow you to trace the process to see which processes have run. If a problem has occurred, double-click on the relevant object to display detailed information on this step.

Follow the Extras • Diagnosis menu path to access analysis transactions that will help you find the reason for partially processed workflows.

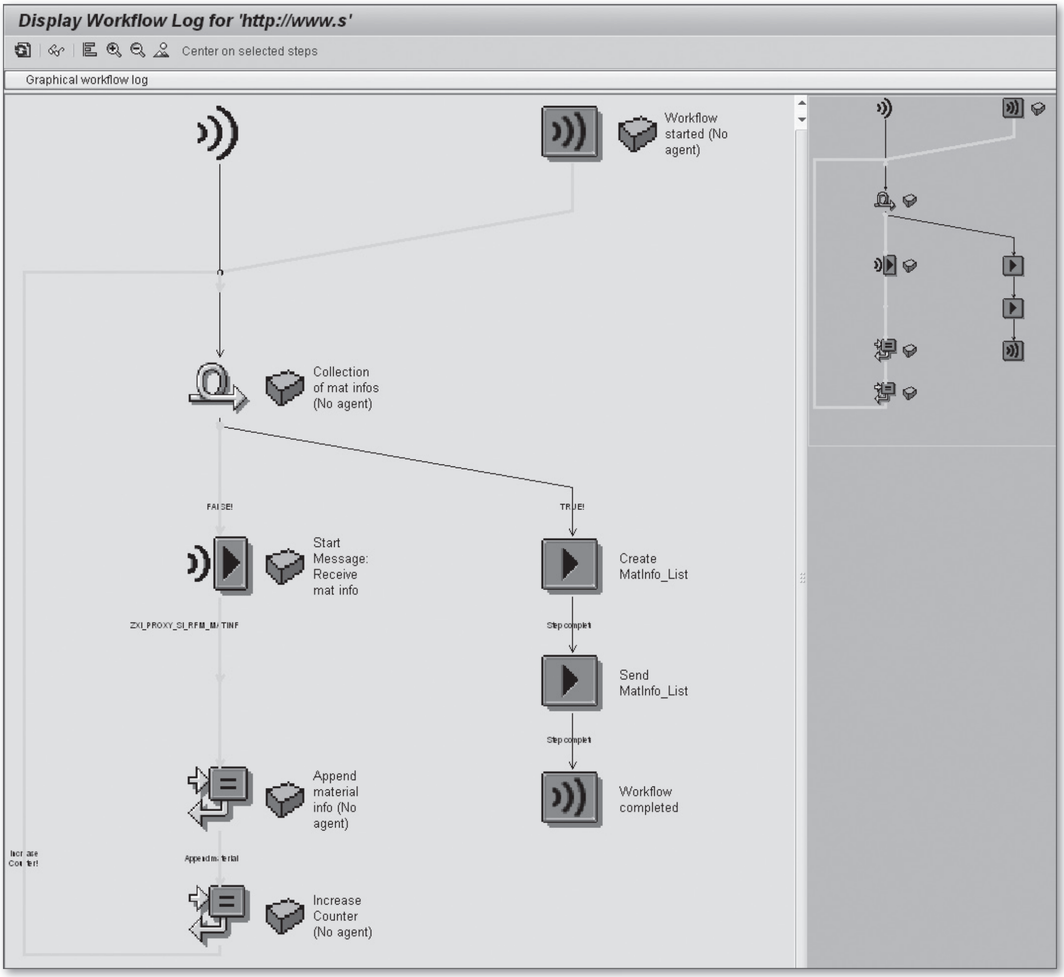


Figure 4.89 Graphical Display of the Workflow Log

4.4.5 Extending the Exercise by Alert Monitoring (Optional)

Using alerts to monitor business processes

The existing integration process can be completed by sending alerts for better control. In the case of this exercise, the alert is a time-out verification, which sends a warning message if the collection loop has not received all three input messages within a specified period. The prerequisite for implementing the subsequent steps is preparing the alert monitoring in Chapter 3.

Creation of the Alert Class

Creating a new alert category

To begin, create an alert category that enables the design of the message and the control of message delivery and visibility. Log on to the PI system, and call Transaction ALRTCDEF to create a new category.[11] Switch to change mode, and open the context menu of the menu path: ALL CLASSIFICATIONS • PI_BOOK.

The `PI_BOOK` category classification was set up during the preparations for the exercise. Click on the list to the right of the directory tree, and then click the CREATE ALERT CATEGORY icon (which looks like a blank page) to create a new category within the classification. Enter the new category, `PI_Alert_##`, in the newly created line, and the name `PI_Alert_##` and a meaningful description. Confirm your entries with the `Enter` key.

Specification of the possible alert recipients

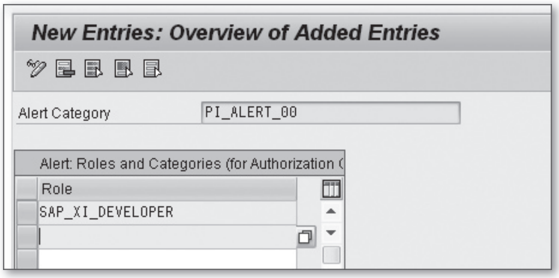
The lower area of the screen now accepts input, and you can determine the characteristics of the category. First, turn on the Dynamic Text option, which passes the text (which you will define later in the integration process) into the alert. You now have two alternatives to determine who should later receive alerts of this category:

- You can enter a fixed list of recipients by clicking the Fixed Recipients icon and entering the appropriate usernames; these users will always receive the related alerts in the Runtime Workbench. This fixed allocation is very inflexible, and, in using it, you take the risk that alerts will not be delivered to all relevant recipients, or that users will be notified inadvertently (due to mistakes in list maintenance).

11 You can also create an alert category using the Alert Configuration option in the Runtime Workbench; however, only the named transaction is called in the Web GUI.

Subscription for alert categories

- A more flexible alternative is to let the users decide whether or not they want to receive an alert of this category. Users can subscribe to alerts in this category in the Runtime Workbench. This subscription option, however, is accessible only for users with certain technical user roles. You can specify the list of relevant roles using the SUBSCRIPTION AUTHORIZATION icon. You can offer the subscription of your own category to all participants by entering the role `SAP_XI_DEVELOPER` (see Figure 4.90). Regardless of the choice of an alternative, you must save your entries and assign a transport request if necessary.



Alert Category: PI_ALERT_00

Alert: Roles and Categories (for Authorization (

Role
SAP_XI_DEVELOPER

Figure 4.90 Entering a Role for the Subscription of an Alert Category

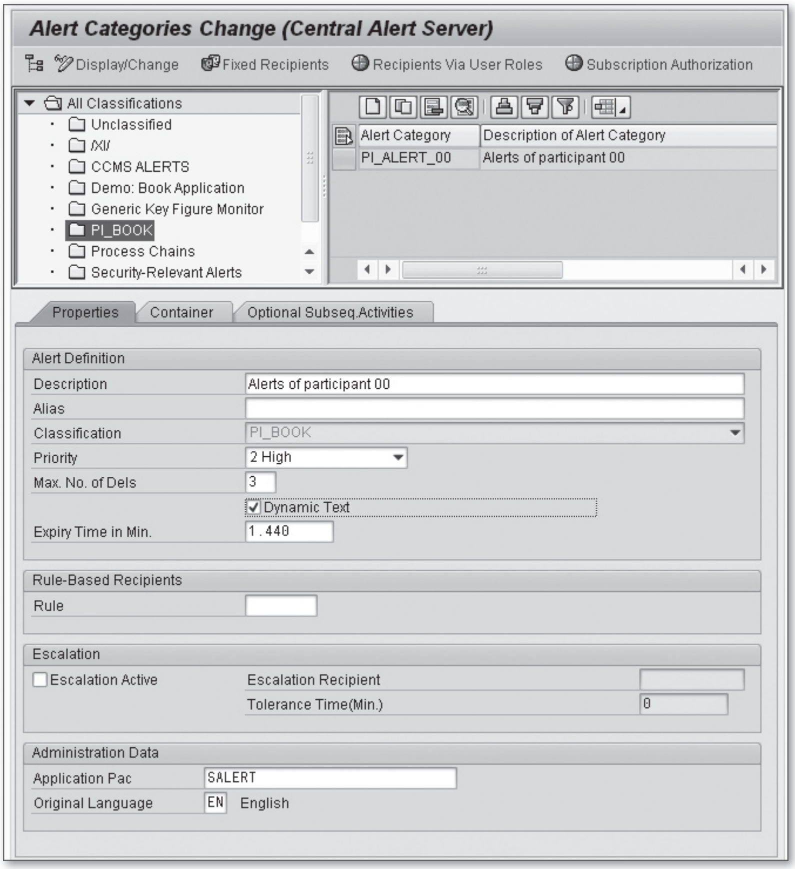
The newly created alert category should now look something like Figure 4.91. Save the new category, and enter your transport request on demand.

Enhancement of the Integration Process

Enhancing the business process with parallel processing

To send an alert, the integration process has to be enhanced with a step sequence that starts in parallel to the Receive step. This parallel sequence sends an alert after a certain amount of time.

Start the Enterprise Services Repository, open your integration process `IP_MatInfo_##`, and then switch to change mode. For a better overview of the process and better transparency in monitoring, create a new block called REGULAR PROCESSING. Move all existing steps into this block, preserving their order. Then drag a Fork step in front of the newly created block, and name it TIMEOUT CONSTRUCT. Then drag the created block to one of the two branches of the Fork step created by default (see Figure 4.92).



Alert Categories Change (Central Alert Server)

Display/Change | Fixed Recipients | Recipients Via User Roles | Subscription Authorization

- All Classifications
  - Unclassified
  - /XI/
  - CCMS ALERTS
  - Demo: Book Application
  - Generic Key Figure Monitor
  - PI_BOOK
  - Process Chains
  - Security-Relevant Alerts

Alert Category	Description of Alert Category
PI_ALERT_00	Alerts of participant 00

Properties | Container | Optional Subseq.Activities

Alert Definition

Description	Alerts of participant 00
Alias	
Classification	PI_BOOK
Priority	2 High
Max. No. of Dels	3
	✓ Dynamic Text
Expiry Time in Min.	1.440

Rule-Based Recipients

Rule	

Escalation

☐ Escalation Active	Escalation Recipient	
	Tolerance Time(Min.)	0

Administration Data

Application Pac	SALERT
Original Language	EN English

Figure 4.91 Properties of the New Alert Category

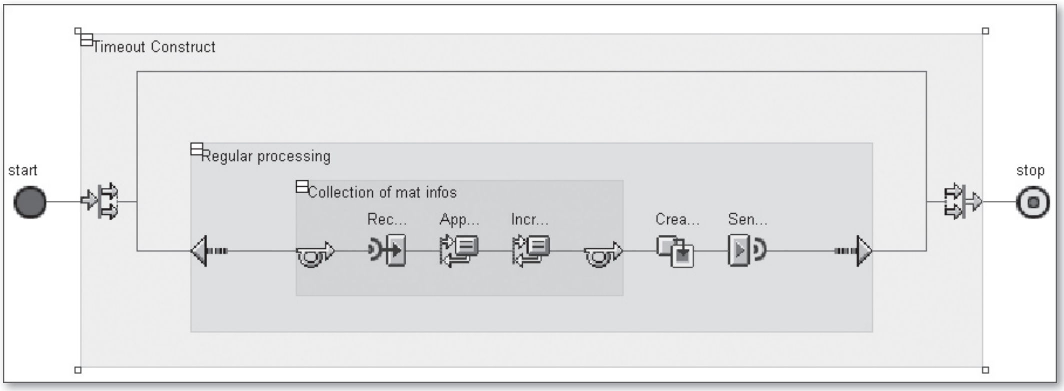


Figure 4.92 Moving the Existing Steps to a Fork Branch

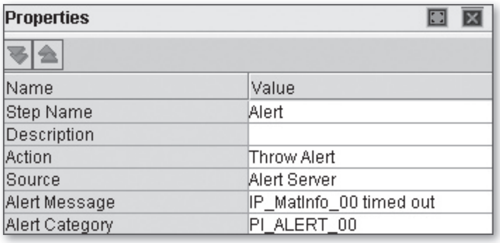
Now drag a block step to the second, previously unused branch, and then name the block Timeout Check. Drag a Wait step into this new block, and then name it Timeout. Select the Wait Specified Time Period option as Type, and enter a wait time of three minutes; this delays the execution of the next steps by three minutes. (This short duration is only suitable for this exercise; in practical use, it will usually be higher.)

Inserting a Wait step

Insert a Control step after the Wait step, and call it Timeout Alert. This step will send an alert when the entire integration process runs longer than three minutes. Set the ACTION property of the control to the THROW ALERT option, thus sending an alert, while the process continues.

Throwing the alert

Set the ALERT SERVER as the SOURCE. Now enter the newly created ALERT CATEGORY following the PI_ALERT_## schema. In the ALERT MESSAGE field, you can finally enter a message that will be displayed in the alert. It is best to choose a meaningful alert, such as IP_MATINFO_## TIMED OUT (Figure 4.93).


Properties

Name	Value
Step Name	Alert
Description	
Action	Throw Alert
Source	Alert Server
Alert Message	IP_MatInfo_00 timed out
Alert Category	PI_ALERT_00

Figure 4.93 Properties of the Control Step in the Integration Process

The complete integration process should now match Figure 4.94. Save, check, and activate the enhanced integration process.

Subscription to the Alert Category and Testing of the Process

If you assigned fixed users when creating the new alert category, it is not necessary to subscribe to a category; continue with the testing of the modified process.

Fixed recipients

To subscribe to an alert category, open the Runtime Workbench and select Alert Inbox from the top menu. Click on Subscription in the blue menu bar on the right, which results in another window that shows all of the alert categories that you can subscribe to, and their subscription status.

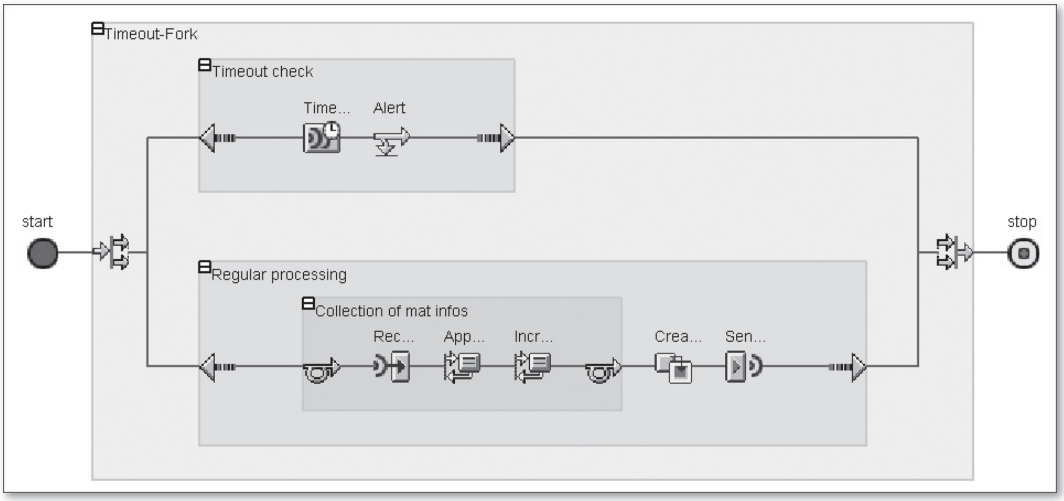
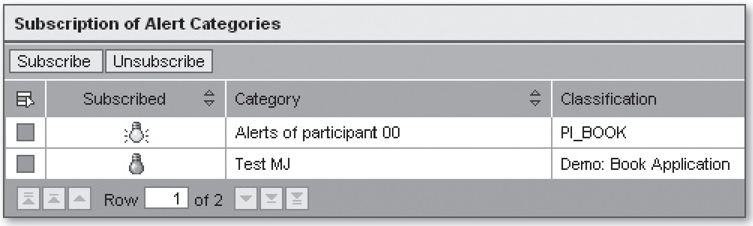


Figure 4.94 Enhanced Integration Process with Time-out Branch

A gray light bulb in the SUBSCRIBED column INDICATES a category that you have not subscribed to, while a bright yellow light bulb indicates an active subscription category (see Figure 4.95). By clicking on the light bulb icon or the corresponding buttons above the list, you can activate or deactivate a subscription. The new settings apply immediately and don't need to be stored separately.



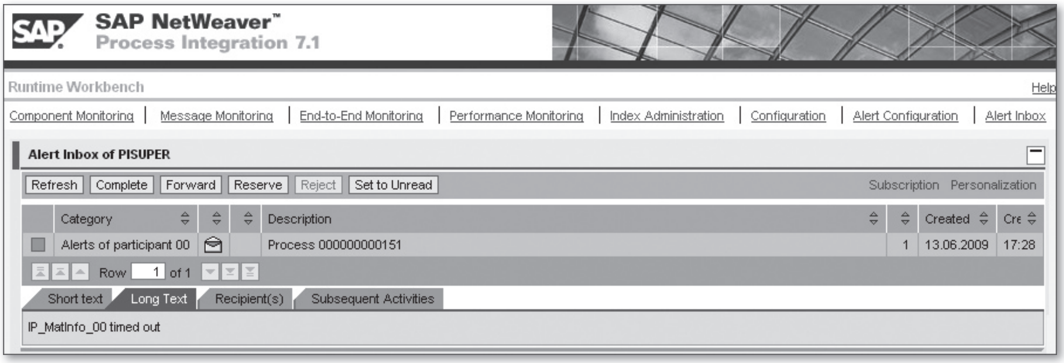
	Subscribed	Category	Classification
		Alerts of participant 00	PI_BOOK
		Test MJ	Demo: Book Application

Figure 4.95 Subscribing to an Assigned Alert Category in the Runtime Workbench

Start of the integration process

To test the enhanced integration process, log on to system A with your user `SYS_A-##`, and call Transaction SE38 to run the program `Z_PROG_MATERIALINFO_##`, as you did in Exercise 1. Enter any material number, and then run the program. Then wait the time specified in the Wait step of the integration process, and call the Runtime Workbench.

Navigate to the Alert Inbox to look at the incoming alert message. Select the message at the beginning of the line to see all of the information. On the SHORT TEXT tab, you can see the automatically inserted workflow instance number, which helps you to identify and analyze the instance causing this alert. On the LONG TEXT tab, you can find the test you entered in the Alert Message of the Control step (see Figure 4.96).


SAP NetWeaver™
Process Integration 7.1
Runtime Workbench
Help
Component Monitoring | Message Monitoring | End-to-End Monitoring | Performance Monitoring | Index Administration | Configuration | Alert Configuration | Alert Inbox
Alert Inbox of PISUPER
Refresh Complete Forward Reserve Reject Set to Unread
Subscription Personalization
Row 1 of 1
Short text Long Text Recipient(s) Subsequent Activities

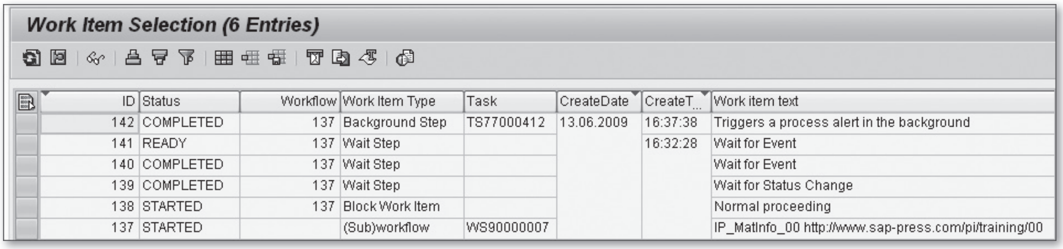
	Category			Description		Created	Cre
	Alerts of participant 00			Process 000000000151	1	13.06.2009	17:28

IP_MatInfo_00 timed out

Figure 4.96 Sent Alert Message in the Alert Inbox of the Runtime Workbench

Monitoring the integration process

Finally, log on to the PI system, and call Transaction SXMB_MONI_BPE to display the workflow monitoring. Follow the BUSINESS PROCESS ENGINE • MONITORING • PROCESS SELECTION menu path, and start this view. Limit the selection period and execute the selection. A list of all work items is displayed; you can see that two Wait steps have been processed and another is ready. One of the processed Wait steps has received the first message, while the second waited the specified time and then sent an alert message. In addition, sending alerts is listed as a separate background step (see Figure 4.97).

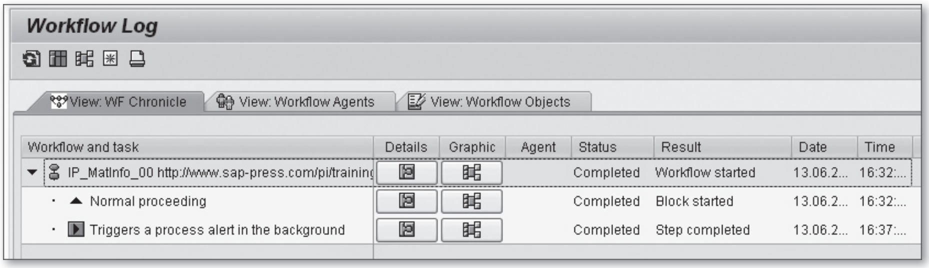

Work Item Selection (6 Entries)

ID	Status	Workflow	Work Item Type	Task	CreateDate	CreateT...	Work item text
142	COMPLETED	137	Background Step	TS77000412	13.06.2009	16:37:38	Triggers a process alert in the background
141	READY	137	Wait Step			16:32:28	Wait for Event
140	COMPLETED	137	Wait Step				Wait for Event
139	COMPLETED	137	Wait Step				Wait for Status Change
138	STARTED	137	Block Work Item				Normal proceeding
137	STARTED		(Sub)workflow	WS90000007			IP_MatInfo_00 http://www.sap-press.com/pi/training/00

Figure 4.97 Work Item View after the Alert Message

Displaying the workflow log

If you jump to the workflow log by double-clicking any work item and then clicking the LOG icon, you can distinguish the two branches (see Figure 4.98).



Workflow and task	Details	Graphic	Agent	Status	Result	Date	Time
IP_MatInfo_00 http://www.sap-press.com/pi/trainin				Completed	Workflow started	13.06.2...	16:32:...
Normal proceeding				Completed	Block started	13.06.2...	16:32:...
Triggers a process alert in the background				Completed	Step completed	13.06.2...	16:37:...

Figure 4.98 Workflow Log of the Enhanced Integration Process after the Alert Message

4.5 Exercise 5: File-to-JDBC

Course of Exercise 5

In Section 4.2, you imported data from a file into business system B using IDoc technology. With SAP NetWeaver PI, you can also write data or messages directly into a database; this is done via the JDBC adapter, which we'll discuss here. To do this, you must first select the corresponding target system (i.e., business system B or a database), which depends on the prefix of the material number. Figure 4.99 shows a schema of this exercise with synchronous communication.

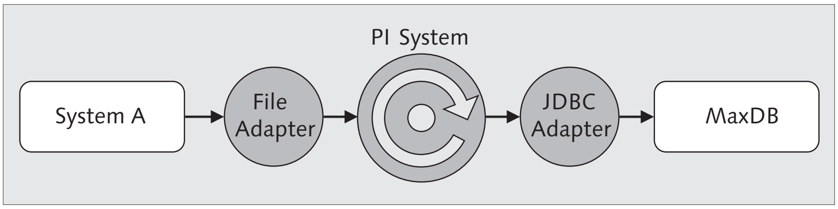


Figure 4.99 Schema of Exercise 5 – File-to-JDBC

4.5.1 Basics

The JDBC adapter allows direct access to database tables, and can be used for both read and write access. This data exchange can take place synchronously or asynchronously.

Driver installation

Before using the JDBC adapter, the installation of a manufacturer-specific driver is required. The individual JAR files can be downloaded from the database vendors and grouped together in an archive called *com.sap.aii.adapter.lib.sda*, so that they can be deployed on the SAP NetWeaver Application Server (AS) using the *Java Support Package Manager* (JSPM). The file extension SDA stands for *software deployment archive*. You can find a detailed guide for the packaging and deployment of driver files in the configuration guide for SAP NetWeaver PI, and also in the SAP Help Portal: *Providing External Drivers for the JDBC and JMS Adapters*.[12]

Configuration of the JDBC adapter

If the JDBC adapter is configured as a sender, the communication channel contains a SQL query that is performed to get relevant data. The retrieved table rows are converted to a message type, which must exist as a design object. After selecting and converting, a SQL update can be performed; this is often used to mark the read-in line with an appropriate flag in a particular column, which prevents these lines from being reread during the next query.

A receiving JDBC adapter can perform various actions on the rows of a table: `UPDATE`, `INSERT`, `DELETE`, `SELECT`, and `UPDATE_INSERT` (the last of which is equivalent to `MODIFY)`. In addition, stored procedures can be performed to directly call more complex SQL statements. The table and action to be performed are determined via the message type of the receiving side.

For this exercise, you need a database of your choice that is accessible from the PI server via a network. In this example, a MaxDB database is used. The material master record from Exercise 2 is written to a database table that corresponds exactly to that structure. A SQL script for the creation of the database can be found in Appendix A.

4.5.2 Design

Overview of the design objects

To write to a database with the JDBC adapter, you must map the contents of the message in XML SQL format. This format is internally

12 You can find these guides at: *http://help.sap.com/saphelp_nwpi71/helpdata/en/33/e6fb40f17af66fe10000000a1550b0/frameset.htm.*

converted by the JDBC adapter, and forwarded to the configured database driver. In this example, it is limited to the INSERT operation.[13] The objects that need to be created for this exercise are shown in Table 4.14.

Object Type	Sender Side	Receiver Side
Service interface		SI_INSERT_SQL_Async_in
Message type		MT_INSERT_SQL
Data type		DT_INSERT_SQL
Operation mapping	OM_SI_Material_Async_Out_to_SI_INSERT_SQL_Async_in	
Message mapping	MM_MT_Material_to_MT_INSERT_SQL	

Table 4.14 Elements in the Enterprise Services Repository for the Exercise File-to-JDBC

Creation of the Interface Objects

Create a new data type in the Enterprise Services Repository. Navigate to the context menu of the Data Types path in your namespace, and create the data type DT_INSERT_SQL. To perform an Insert statement, this data type must conform to the XML SQL format of the INSERT command. The command has the format shown in Listing 4.3.

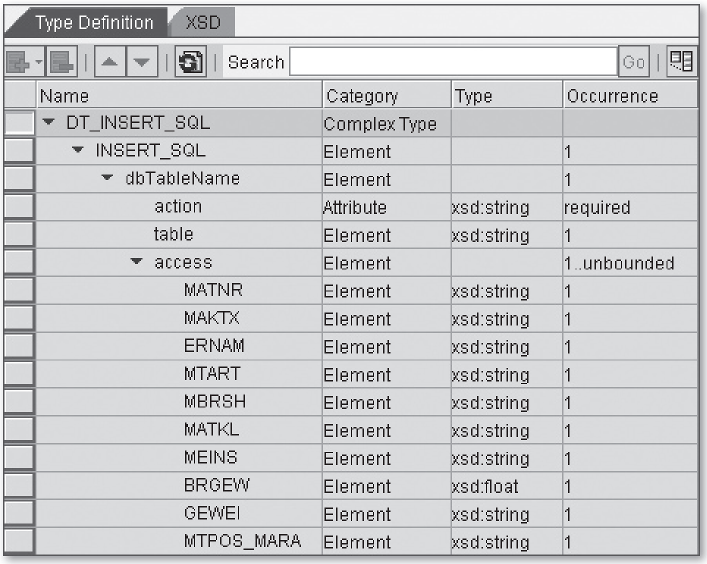
Within the <access> blocks, you can specify the data you want to insert in the table; the XML tags correspond to the column names of your database table. This statement must contain at least one <access> element. The data type in our example is shown in Figure 4.100.

13 You can find additional information on XML SQL format in the SAP Help Portal at: *http://help.sap.com/saphelp_nwpi71/helpdata/EN/2e/96fd3f2d14e869e10000000a155106/frameset.htm.*

Creation of the interface objects

```
<root>
  <StatementName>
    <dbTableName action="INSERT">
      <table>realDbTableName</table>
      <access>
        <col1>val1</col1>
        <col2>val2</col2>
      </access>
      <access>
        <col1>val11</col1>
      </access>
    </dbTableName>
  </StatementName>
</root>
```

Listing 4.3 XML Example for the XML SQL INSERT Operation

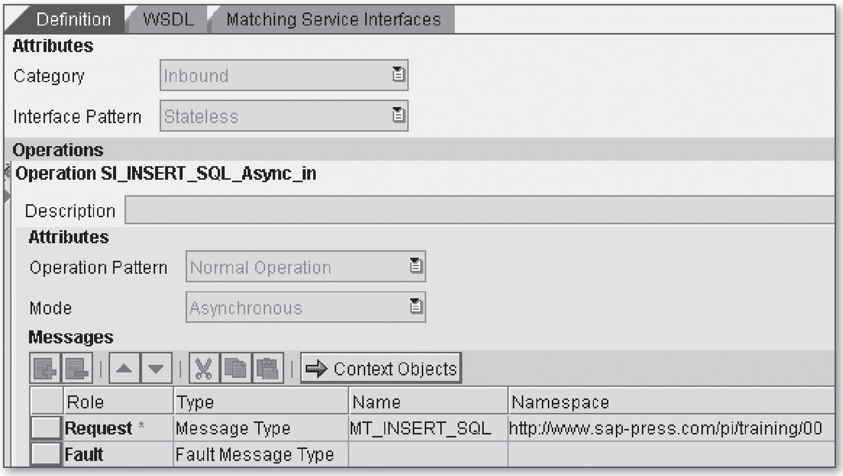


Name	Category	Type	Occurrence
DT_INSERT_SQL	Complex Type		
INSERT_SQL	Element		1
dbTableName	Element		1
action	Attribute	xsd:string	required
table	Element	xsd:string	1
access	Element		1..unbounded
MATNR	Element	xsd:string	1
MAKTX	Element	xsd:string	1
ERNAM	Element	xsd:string	1
MTART	Element	xsd:string	1
MBRSH	Element	xsd:string	1
MATKL	Element	xsd:string	1
MEINS	Element	xsd:string	1
BRGEW	Element	xsd:float	1
GEWEI	Element	xsd:string	1
MTPOS_MARA	Element	xsd:string	1

Figure 4.100 Type Definition for the Scenario File-to-JDBC

Now create the `MT_INSERT_SQL` message type, based on the newly created data type. To do this, open the context menu of the Message Types path in your namespace, and create a new message type. As a data type, enter the `DT_INSERT_SQL` data type that you just created.

Finally, the service interface for this scenario has to be created; an asynchronous inbound interface is required. Create the `SI_INSERT_SQL_Async_in` service interface via the context menu of the Service Interfaces path, and reference the `MT_INSERT_SQL` message type as the request message type. The settings of the service interface are shown in Figure 4.101.


Definition | WSDL | Matching Service Interfaces

Attributes

Category: Inbound

Interface Pattern: Stateless

Operations

Operation SI_INSERT_SQL_Async_in

Description:

Attributes

Operation Pattern: Normal Operation

Mode: Asynchronous

Messages

Role	Type	Name	Namespace
Request *	Message Type	MT_INSERT_SQL	http://www.sap-press.com/pi/training/00
Fault	Fault Message Type		

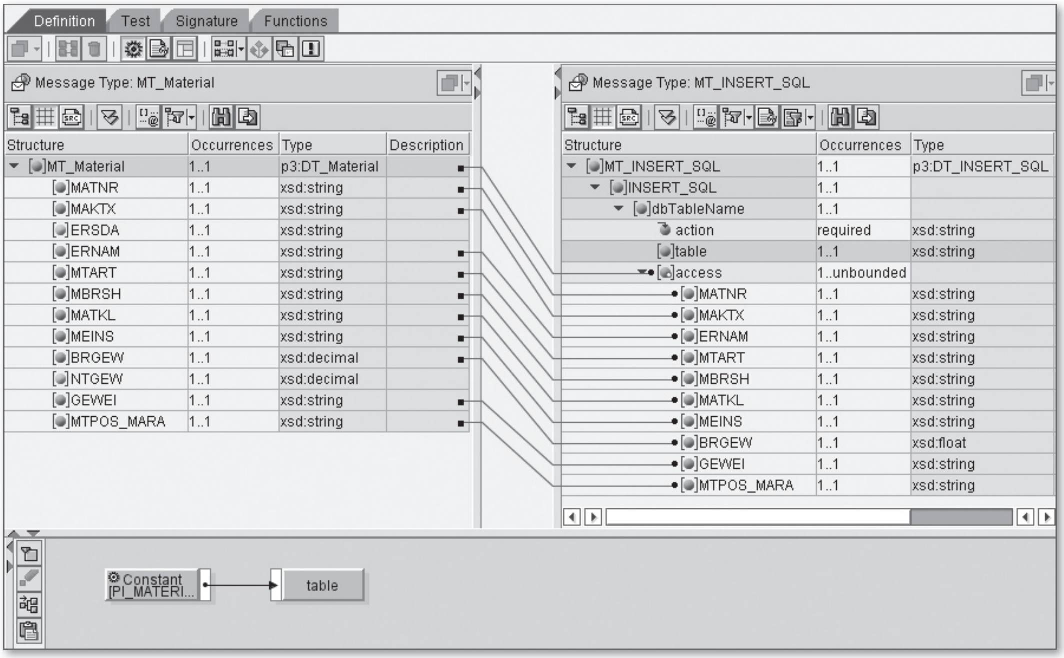
Figure 4.101 Service Interface for the Scenario File-to-JDBC

Creation of the Mapping Objects

For the enhancement of the scenario from Section 4.2, you still need a mapping between the sending file format and the newly created SQL-XML format. Create a new message mapping with the name `MM_MT_Material_to_MT_INSERT_SQL`, using the context menu of the Message Mappings path in your namespace mapping. For the left side of the mapping, enter `MT_Material` as the message type; for the right side, `MT_INSERT_SQL`.

Creating the mapping objects

Map the `action` and `table` elements in the target message to the constant value `INSERT`, and the database name in which you want to write the values. In this example, the database name is `PI_MATERIAL`. Link the elements of the ACCESS node with the corresponding fields of the source message (see Figure 4.102). Test the mapping, and check whether the same XML message structure is produced as in the example XML for the XML SQL Insert command.



Structure	Occurrences	Type	Description
MT_Material	1..1	p3:DT_Material	
MATNR	1..1	xsd:string	
MAKTX	1..1	xsd:string	
ERSDA	1..1	xsd:string	
ERNAM	1..1	xsd:string	
MTART	1..1	xsd:string	
MBRSH	1..1	xsd:string	
MATKL	1..1	xsd:string	
MEINS	1..1	xsd:string	
BRGEW	1..1	xsd:decimal	
NTGEW	1..1	xsd:decimal	
GEWEI	1..1	xsd:string	
MTPOS_MARA	1..1	xsd:string	

Structure	Occurrences	Type
MT_INSERT_SQL	1..1	p3:DT_INSERT_SQL
INSERT_SQL	1..1	
dbTableName	1..1	
action	required	xsd:string
table	1..1	xsd:string
access	1..unbounded	
MATNR	1..1	xsd:string
MAKTX	1..1	xsd:string
ERNAM	1..1	xsd:string
MTART	1..1	xsd:string
MBRSH	1..1	xsd:string
MATKL	1..1	xsd:string
MEINS	1..1	xsd:string
BRGEW	1..1	xsd:float
GEWEI	1..1	xsd:string
MTPOS_MARA	1..1	xsd:string

Figure 4.102 Message Mapping of the Exercise File-to-JDBC

As a final step of the design, the operation mapping is now missing. Create it with the name `OM_SI_Material_Async_Out_to_SI_INSERT_SQL_Async_in`, using the context of the Operation Mappings path in your namespace. Define the service interface `SI_Material_Async_Out` as the output interface, and `SI_INSERT_SQL_Async_in` as the target interface. By clicking on the Read Operations button, the corresponding message types of the interface are loaded. Now you can set the message mapping you just created as the mapping program. Activate all of the design objects and switch to the Integration Directory.

4.5.3 Configuration

For this exercise, you don't create a new scenario, but expand the scenario from Section 4.2, such that the target is determined depending on the material number's prefix in the incoming message. If the material number's prefix is *db*, the material is written to the database; otherwise, the IDoc that has already been configured in Section 4.2 is created and posted to business system B.

Overview of the configuration objects

The configuration objects that you will create for this exercise are shown in Table 4.15. The receiver determination does not need to be changed for the enhancement, as it already exists. The existing interface determination must be adjusted so that the message will be routed depending on the prefix of the material number.

Type of Object	Sender Side: System A	Receiver Side: System B
Communication channel		`JDBC_ReceiverChannel_##`
Receiver agreement		`|SystemA||SystemB |SI_INSERT_SQL_Async_in`
Receiver determination		
Interface determination		

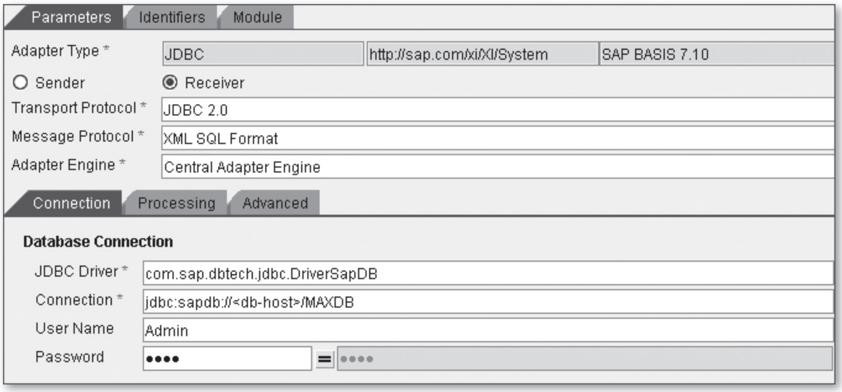
Table 4.15 Elements in the Integration Directory for the Exercise File-to-JDBC

Open the context menu of the Communication Channel node in the Integration Builder, and create a new communication channel with the name `JDBC_ReceiverChannel_##`. Select business system B as the communication component.

Configuration of the JDBC adapter

The adapter is of type JDBC. The communication channel is created as a RECEIVER. Make sure that as MESSAGE PROTOCOL the XML SQL FORMAT is set. Alternatively, you can set the Native SQL String option. This protocol is mainly intended for testing purposes.

In the CONNECTION tab, enter the connection to the database. The values of the parameters JDBC DRIVER and CONNECTION are dependent on the database used. Enter a user for the database in the USER NAME and PASSWORD fields; this user must have the authorization to execute an `INSERT` command on the appropriate table (see Figure 4.103).



Adapter Type *	JDBC	http://sap.com/xi/XI/System	SAP BASIS 7.10
○ Sender	◉ Receiver		
Transport Protocol *	JDBC 2.0		
Message Protocol *	XML SQL Format		
Adapter Engine *	Central Adapter Engine		

Database Connection

JDBC Driver *	com.sap.dbtech.jdbc.DriverSapDB
Connection *	jdbc:sapdb://<db-host>/MAXDB
User Name	Admin
Password	•••• = ••••

Figure 4.103 Settings of the JDBC Receiver Channel for System B

To control the scenario depending on the prefix of the material number, expand the interface determination `|SystemA|SI_Material_Async_out|SystemB|` by another configured receiver interface. Make sure that the new receiver interface comes first, and that the Maintain Order At Runtime checkbox is set. In doing so, the condition of the new receiver interface is checked. If the condition is not met, an IDoc is sent to business system B.

Configuration of the interface determination

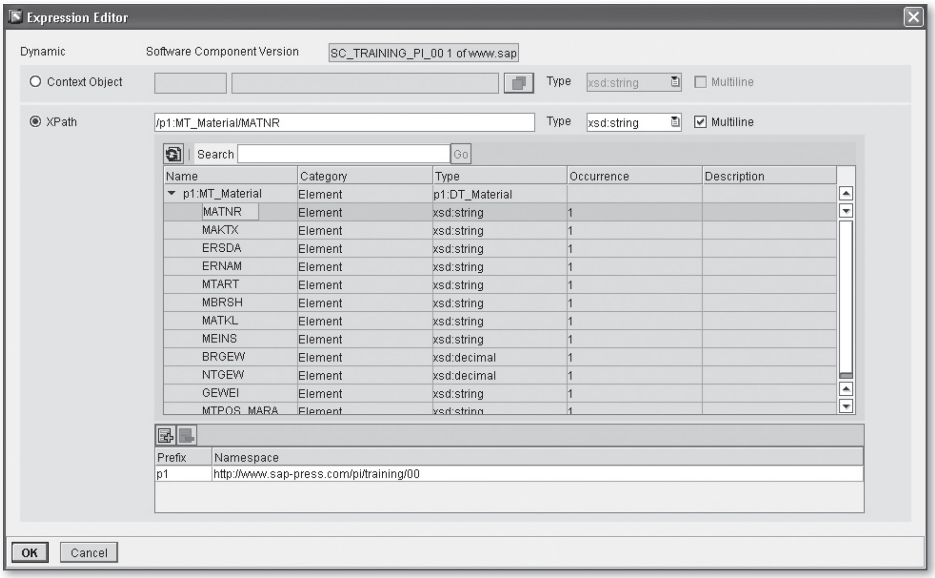
Select the `SI_INSERT_SQL_Async_in` service interface, and specify the operation mapping as `OM_SI_Material_Async_Out_to_SI_INSERT_SQL_Async_in`. Finally, specify the desired condition in the condition editor. Start the condition editor using the values help of the Condition field.

The condition editor will open in a new window, and allows you to perform an examination of the incoming message's content. Using the value help in the Left Operand field, you get to the expression editor that allows you to specify the field you want to check (see Figure 4.104).

Configuration using the expression editor

Select the `MATNR` field, and then click OK. Return to the window of the condition editor and select ≈ as an operator. This operator can check a string for a pattern, and you can use the following wildcards:

- + for any character
- * for any character sequence


Expression Editor

Dynamic   Software Component Version SC_TRAINING_PI_00 1 of www.sap

○ Context Object   Type xsd:string   ☐ Multiline

● XPath /p1:MT_Material/MATNR   Type xsd:string   ☑ Multiline

Search   Go

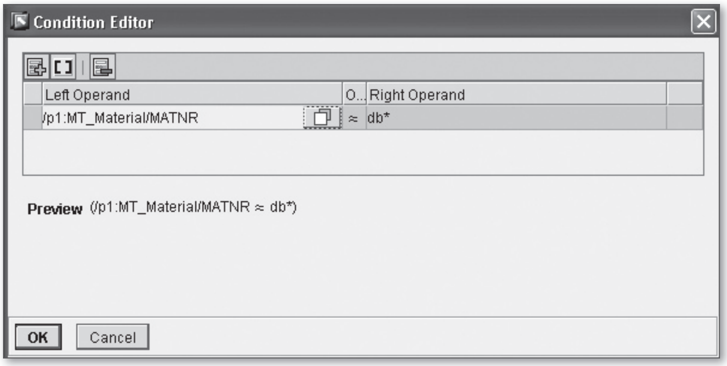
Name	Category	Type	Occurrence	Description
p1:MT_Material	Element	p1:DT_Material		
MATNR	Element	xsd:string	1	
MAKTX	Element	xsd:string	1	
ERSDA	Element	xsd:string	1	
ERNAM	Element	xsd:string	1	
MTART	Element	xsd:string	1	
MBRSH	Element	xsd:string	1	
MATKL	Element	xsd:string	1	
MEINS	Element	xsd:string	1	
BRGEW	Element	xsd:decimal	1	
NTGEW	Element	xsd:decimal	1	
GEWEI	Element	xsd:string	1	
MTPOS_MARA	Element	xsd:string	1	

Prefix	Namespace
p1	http://www.sap-press.com/pi/training/00

OK   Cancel

Figure 4.104 Expression Editor

Enter the string `db*` in the field of the right operand (see Figure 4.105); as a result, all materials that have a material number starting with the prefix db are forwarded to the database. All others will be sent to business system B.


Condition Editor

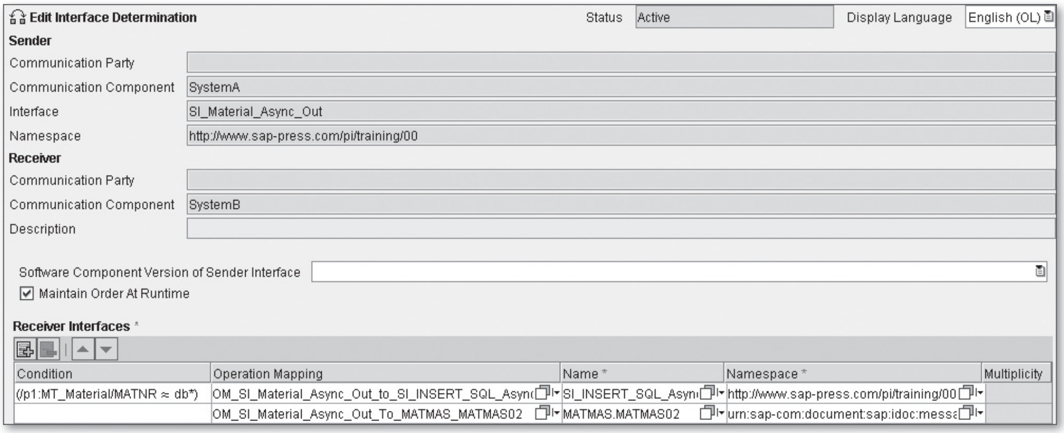
Left Operand	O...	Right Operand
/p1:MT_Material/MATNR	≈	db*

Preview (/p1:MT_Material/MATNR ≈ db*)

OK   Cancel

Figure 4.105 Condition Editor

After setting all of the parameters, save the interface determination (see Figure 4.106).



Edit Interface Determination — Status: Active — Display Language: English (OL)

Sender

Communication Party	
Communication Component	SystemA
Interface	SI_Material_Async_Out
Namespace	http://www.sap-press.com/pi/training/00

Receiver

Communication Party	
Communication Component	SystemB
Description	

Software Component Version of Sender Interface

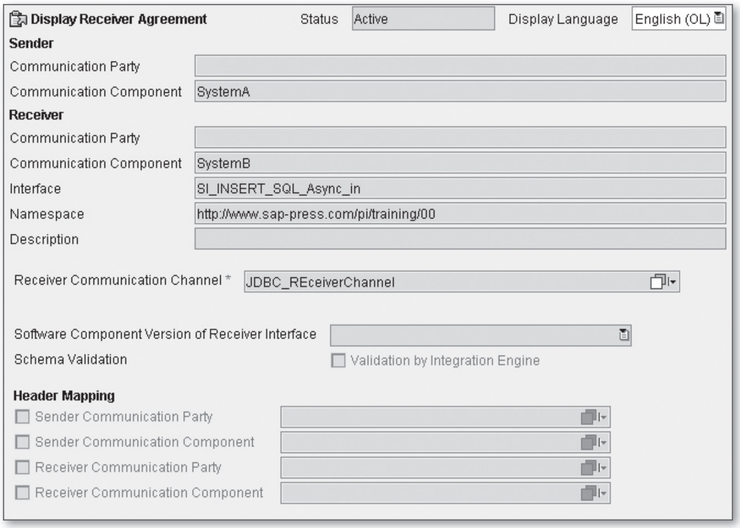
☑ Maintain Order At Runtime

Receiver Interfaces *

Condition	Operation Mapping	Name *	Namespace *	Multiplicity
(/p1:MT_Material/MATNR ≈ db*)	OM_SI_Material_Async_Out_to_SI_INSERT_SQL_Asyn	SI_INSERT_SQL_Asyn	http://www.sap-press.com/pi/training/00	
	OM_SI_Material_Async_Out_To_MATMAS_MATMAS02	MATMAS.MATMAS02	urn:sap-com:document:sap:idoc:mess	

Figure 4.106 Interface Determination of the Scenario File-to-JDBC

Finally, the receiver agreement for the JDBC receiver still needs to be created. You can create it via the context menu of the Receiver Agreement node. Select business system A as the sending communication component, and business system B as the recipient. Enter SI_INSERT_SQL_Async_in as the receiver interface. Configure the newly created receiver channel JDBC_ReceiverChannel_## as the communication channel of this object (see Figure 4.107). Save and activate all of the configuration objects.



Display Receiver Agreement — Status: Active — Display Language: English (OL)

Sender

Communication Party	
Communication Component	SystemA

Receiver

Communication Party	
Communication Component	SystemB
Interface	SI_INSERT_SQL_Async_in
Namespace	http://www.sap-press.com/pi/training/00
Description	

Receiver Communication Channel *: JDBC_REceiverChannel

Software Component Version of Receiver Interface

Schema Validation: ☐ Validation by Integration Engine

Header Mapping

☐ Sender Communication Party
☐ Sender Communication Component
☐ Receiver Communication Party
☐ Receiver Communication Component

Figure 4.107 Receiver Agreement for the Exercise File-to-JDBC

4.5.4 Process and Monitoring

The integration scenario in this exercise is triggered the same way as it was in Exercise 2 — by creating *pi_output_#.dat* in the sending adapter's directory. For this example, create a material whose material number (field `MATNR`) starts with the prefix *db*.

Monitoring of Exercise 5

To create and place the file, you can use either the template file or the file upload and download program that is provided on the website of this book (*http://www.sap-press.com*). You can also use the file from the first exercise. (It may be necessary to remove the write protection of this file if you are on a UNIX system.)

Depending on which poll interval is set in the communication channel, the file is archived or deleted after a few minutes. In PI monitoring (Transaction SXMB_MONI), you can now check that the correct interface has been identified. Finally, in the database, you can verify whether the material has been created in the table.

If the output message was handed over to the JDBC adapter, the scenario ran successfully from the perspective of SAP NetWeaver PI. The monitor for processed XML messages (Transaction SXMB_MONI) won't show error messages that occur in the adapter or in the database; for this you can use the monitor of the JDBC adapter, which can be found in the Runtime Workbench (*http://<J2EE-Host>:<J2EE-Port>/rwb/index.jsp*).

4.5.5 Alternative Java Mapping (Optional)

Limitations of Java mappings

In the spirit of Java, let's also discuss the server-side Java mapping. The graphical mapping within the Enterprise Services Repository is automatically translated into Java code, but only provides limited possibilities for mapping. Experienced Java users may prefer to create extensive Java code for mapping in their favorite editor; however, Java mapping can't be stateful. In other words, you can't write any data in database tables, because you cannot automatically exclude double entries.[14]

14 Detailed information about the limitations and possibilities of Java mappings in SAP NetWeaver PI can be found at: *http://help.sap.com/saphelp_nwpi71/helpdata/en/e2/e13fcd80fe47768df001a558ed10b6/frameset.htm*.

Creation of the Java Mapping

SAP NetWeaver Developer Studio for Java mappings

The Java mapping, which will later be imported into the Enterprise Services Repository and thus lies in the PI system, consists of a single Java class in a JAR file. In principle, this class can be developed in any editor; however, you must reference SAP Java libraries. To use these libraries directly, and to benefit from error-checking, use the SAP NetWeaver Developer Studio in the same release as the Java stack of the PI system.

Start the SAP NetWeaver Developer Studio, and create a new development project via the FILE • NEW • PROJECT menu path. In the JAVA category, choose the JAVA PROJECT option (see Figure 4.108).

Creation of the class

Name the new project according to the scheme `PI_Java_Mapping_##`, and proceed to the next step in the Creation Wizard. Click on FINISH to generate the project. If the Java Perspective is not already set, confirm the switch to this view. The new project will now appear in the tree on the left side.

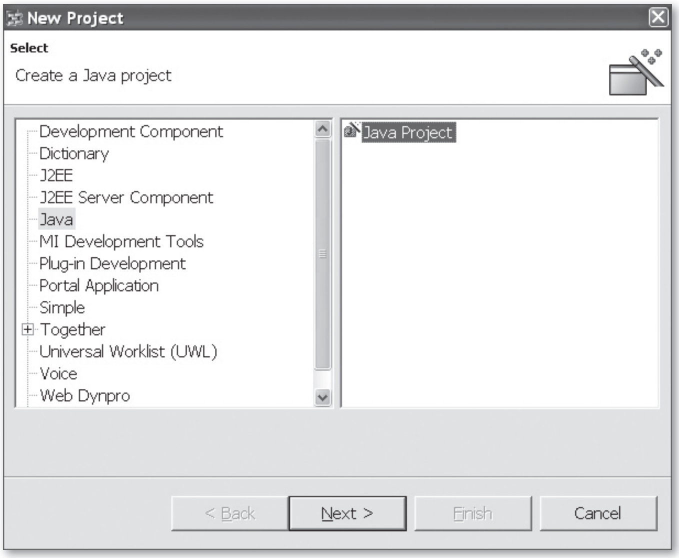
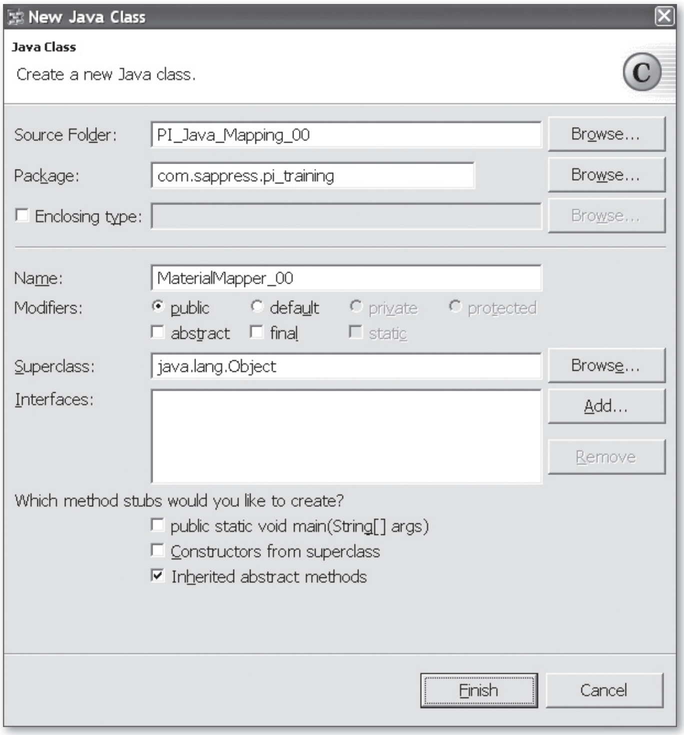


Figure 4.108 Creation of a New Java Project

To create a new class within the project, open the context menu of the new project and follow the NEW • CLASS menu path. Name the package in which the new class is to be created `com.sappress.pi_training`, and enter the new class name `MaterialMapper_##` (see Figure 4.109). Complete the Setup Wizard with the FINISH button.

Integration of the SAP Java libraries for mappings

To access the aforementioned SAP Java libraries within the source code, reopen the context menu of the project, and select the Properties option. In the window, select the Java Build Path directory on the left, and then navigate to the Libraries tab. Click on the Add Variable... button on the right, and select the `SAP_SYSTEM_ADD_LIBS` variable. Then click Extend to see the subtree. Navigate to the path of the first library in Table 4.16, and confirm your choice by clicking OK.



New Java Class

Java Class

Create a new Java class.

Source Folder: PI_Java_Mapping_00

Package: com.sappress.pi_training

☐ Enclosing type:

Name: MaterialMapper_00

Modifiers: [✓ public] default private protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

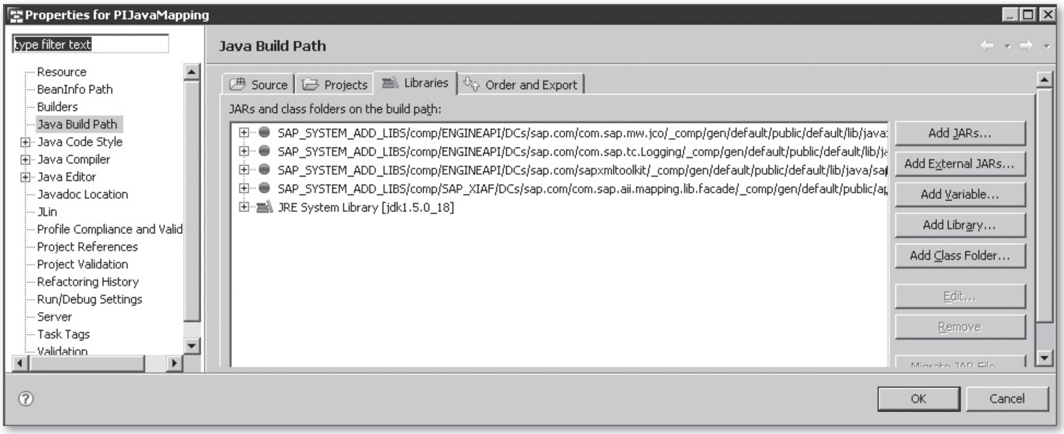
Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☑ Inherited abstract methods

Figure 4.109 Creation of a Class within the Project

Library	Path
Mapping API	COMP • SAP_XIAF • DCs • SAP.COM • COM.SAP.AII.MAPPING.LIB.FACADE • _COMP • GEN • DEFAULT • PUBLIC • API • LIB • JAVA • COM.SAP.AII.MAPPING.API.FILTER.JAR
SAP XML tool kit	COMP • ENGINEAPI • DCs • SAP.COM • SAPXMLTOOLKIT • _COMP • GEN • DEFAULT • PUBLIC • DEFAULT • LIB • JAVA • SAPXMLTOOLKIT
JCo	COMP · ENGINEAPI • DCs • SAP.COM • COM.SAP.MW.JCO • _COMP • GEN • DEFAULT • PUBLIC • DEFAULT • LIB • JAVA • COM.SAP.MW.JCO
Logging	COMP • ENGINEAPI • DCs • COM.SAP.TC.LOGGING • _COMP • GEN • DEFAULT • PUBLIC • DEFAULT • LIB • JAVA • COM.SAP.TC.LOGGING

Table 4.16 SAP Java Libraries for the Realization of the Java Mapping

Follow the same procedure with the three remaining libraries. When finished, the listing of the Java libraries in your project should look similar to Figure 4.110.



JARs and class folders on the build path:

```
SAP_SYSTEM_ADD_LIBS/comp/ENGINEAPI/DCs/sap.com/com.sap.mw.jco/_comp/gen/default/public/default/lib/java:
SAP_SYSTEM_ADD_LIBS/comp/ENGINEAPI/DCs/sap.com/com.sap.tc.Logging/_comp/gen/default/public/default/lib/ja
SAP_SYSTEM_ADD_LIBS/comp/ENGINEAPI/DCs/sap.com/sapxmltoolkit/_comp/gen/default/public/default/lib/java/sa
SAP_SYSTEM_ADD_LIBS/comp/SAP_XIAF/DCs/sap.com/com.sap.aii.mapping.lib.facade/_comp/gen/default/public/ap
JRE System Library [jdk1.5.0_18]
```

Figure 4.110 SAP Java Libraries of the Java Mapping Project

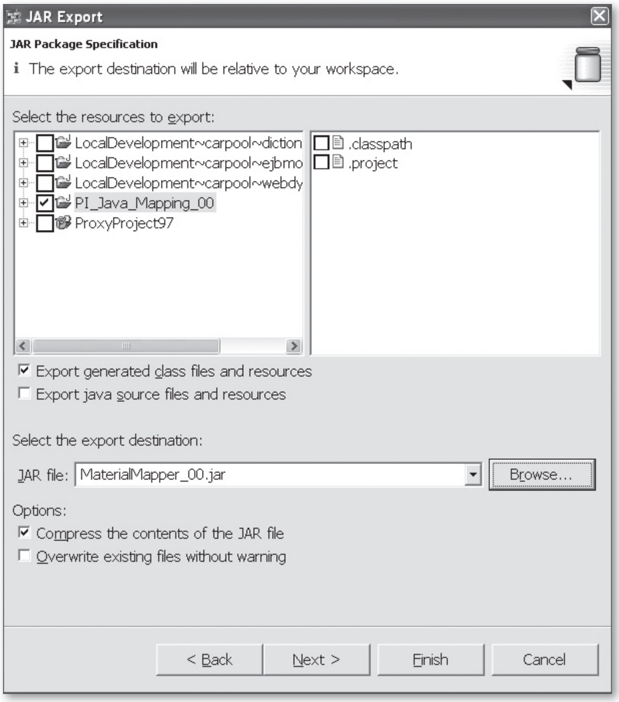
Structure and flow of the class

Now copy the code from Appendix A (or from the template file of the online resources for this book) into the new class. At this point, we will briefly describe the structure and the flow of the class. In the `execute` method, in which the real event takes place, the main message is composed step by step in the form of the new `result` document. First, the message type is declared, and then the table name `PI_MATERIAL` is set.

The `appendElement` method at the end of the coding only serves the simplified appending of elements to the target message. The elements of the source message are then appended to the target message without further transformation. Finally, the target message is returned by the `transform` method.

Export of the JAR archive

You now must create a JAR file out of the project with the new class, which is necessary for importing into the Enterprise Services Repository. To do this, follow the FILE • EXPORT menu path. Select the JAR FILE option and click on NEXT. In the upper left, and on the right, verify that only your project (and no other entry) is selected. Enter the name of the JAR file to be created according to the `MaterialMapper_##.jar` scheme, including the path, and click FINISH (see Figure 4.111).


JAR Export

JAR Package Specification

The export destination will be relative to your workspace.

Select the resources to export:

- [ ] LocalDevelopment~carpool~diction
- [ ] LocalDevelopment~carpool~ejbmo
- [ ] LocalDevelopment~carpool~webdy
- [✓] PI_Java_Mapping_00
- [ ] ProxyProject97

- [ ] .classpath
- [ ] .project

[✓] Export generated class files and resources
[ ] Export java source files and resources

Select the export destination:

JAR file: MaterialMapper_00.jar

Options:

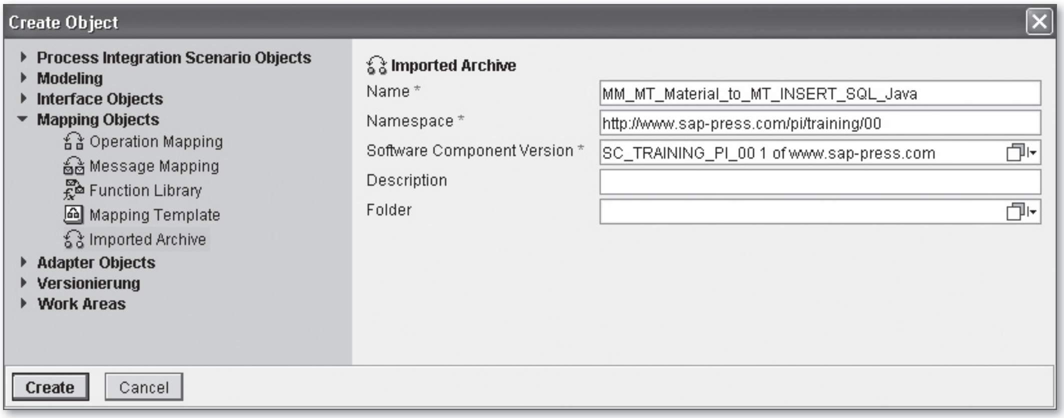
[✓] Compress the contents of the JAR file
[ ] Overwrite existing files without warning

Figure 4.111 Exporting the Project as a JAR File

Integration of the Java Mapping

Creation of an imported archive in the Enterprise Services Repository

Now switch to the Enterprise Services Repository and import the generated JAR archive to your namespace. Follow the OBJECT • NEW menu path, or click on the corresponding icon. On the left side of the Creation Wizard, follow the MAPPING OBJECTS • IMPORTED ARCHIVE menu path, and enter the name `MM_MT_Material_to_MT_INSERT_SQL_Java` (see Figure 4.112). Make sure that your namespace is displayed, and enter a meaningful description before you exit the wizard.



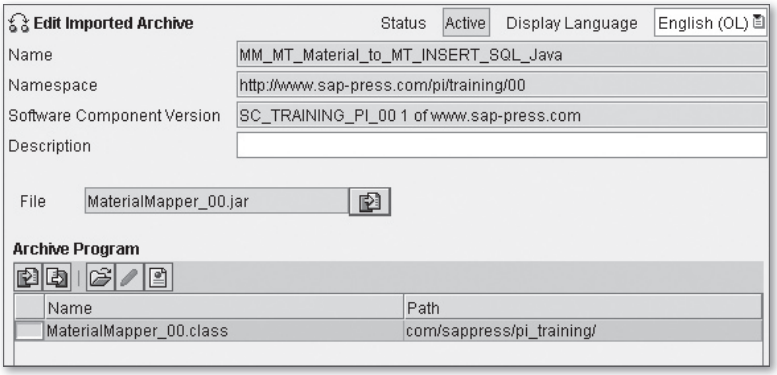
Imported Archive

Name *	MM_MT_Material_to_MT_INSERT_SQL_Java
Namespace *	http://www.sap-press.com/pi/training/00
Software Component Version *	SC_TRAINING_PI_00 1 of www.sap-press.com
Description	
Folder	

Figure 4.112 Creation of an Imported Archive for the Java Mapping in the Enterprise Services Repository

Import of the JAR archive

After creating the new object, the detail window opens, allowing you to import the JAR file. From the FILE field, click on the IMPORT ARCHIVE icon on the right side, and select the *.JAR file type in the Files of type box. Navigate to the JAR archive that you just exported, and then click Open. In the FILE field, the corresponding JAR archive and the relevant files of the archive are now displayed (see Figure 4.113). Save the new object; in doing so, the new Imported Archives directory is created, and appears in the tree on the left side.



Edit Imported Archive — Status: Active — Display Language: English (OL)

Name	MM_MT_Material_to_MT_INSERT_SQL_Java
Namespace	http://www.sap-press.com/pi/training/00
Software Component Version	SC_TRAINING_PI_00 1 of www.sap-press.com
Description	
File	MaterialMapper_00.jar

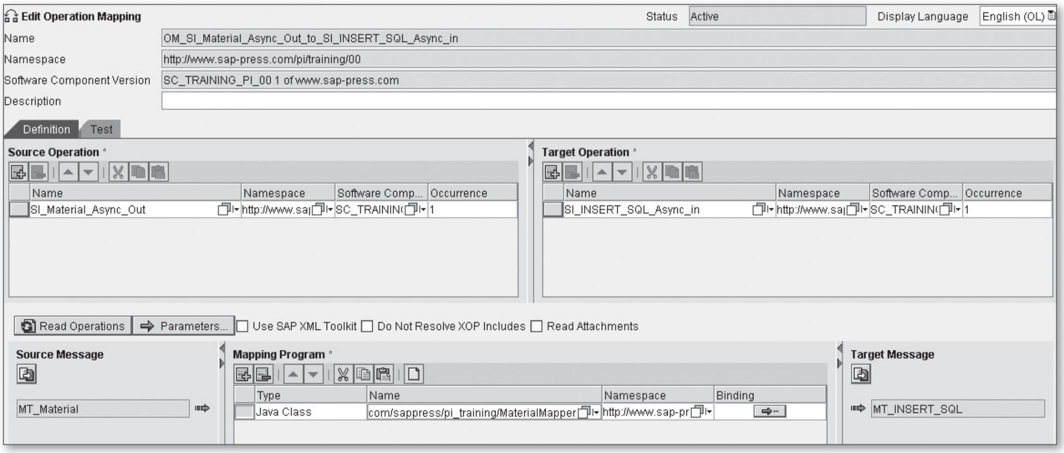
Archive Program

Name	Path
MaterialMapper_00.class	com/sappress/pi_training/

Figure 4.113 Details of the Imported Archive after the Import of the JAR Archive

Adjustment of the operation mapping

Now change to the existing operation mapping `OM_SI_Material_Async_Out_to_SI_INSERT_SQL_Async_in`, and then switch to the change mode. In the Mapping Program section in the bottom-center of the screen, select the Java Class type in the Type column. This allows you to select the imported archive as a mapping program. Then, in the Name field, select the newly imported archive `com/sappress/pi_training/MaterialMapper_##` (see Figure 4.114). Save the adjusted mapping, and test it. Finally, activate the two newly created or modified objects.



Edit Operation Mapping — Status: Active — Display Language: English (OL)

Name	OM_SI_Material_Async_Out_to_SI_INSERT_SQL_Async_in
Namespace	http://www.sap-press.com/pi/training/00
Software Component Version	SC_TRAINING_PI_00 1 of www.sap-press.com
Description	

Definition | Test

Source Operation *

Name	Namespace	Software Comp...	Occurrence
SI_Material_Async_Out	http://www.sa	SC_TRAININ	1

Target Operation *

Name	Namespace	Software Comp...	Occurrence
SI_INSERT_SQL_Async_in	http://www.sa	SC_TRAININ	1

Read Operations | Parameters... | ☐ Use SAP XML Toolkit ☐ Do Not Resolve XOP Includes ☐ Read Attachments

Source Message: MT_Material

Mapping Program *

Type	Name	Namespace	Binding
Java Class	com/sappress/pi_training/MaterialMapper	http://www.sap-pr	

Target Message: MT_INSERT_SQL

Figure 4.114 Replacement of the Existing Mapping Program with the Imported Archive

4.6 More Adapters

Despite the variety of integration examples, not all of the adapters can be discussed in the presented exercises and the following case study. Nonetheless, we provide an overview of the omitted, but still frequently used, adapter types. The focus lies on particular aspects of their real-life usage. (The mail adapter is not considered here because it is implemented in the case study in Chapter 5, SARIDIS Case Study in Sales and Distribution.)

4.6.1 Java Message Service (JMS) Adapter

Application areas of the JMS adapter

The JMS adapter is mainly used for exchanging data with other enterprise application integration (EAI) and messaging systems, such as IBM WebSphere MQ or SonicMQ. In contrast to other adapters, the JMS adapter only allows asynchronous communication.

As with the JDBC adapter, JMS also requires vendor-dependent drivers to be deployed. The drivers can be obtained from the vendors of the integrated EAI systems, and set up the same way as the JDBC drivers. Because the queues of other products are read or written to when using this adapter, you also need to configure the connected EAI product, and the SAP NetWeaver PI communication settings.

Configuration of the JMS adapter

When configuring the JMS adapter both as a sender and a receiver, SAP provides additional transport protocols, some of which are product specific. The settings of the sending JMS adapter lets you establish correlations based on different criteria. Additionally, you can choose whether the JMS payload is transferred as an entire message or as a message payload. For asynchronous communication, there are two qualities of service: exactly once (EO) and exactly-once-in-order (EOIO).

In its role as a receiver, the JMS adapter allows you to return the parameters and to specify the validity period and the priority of the JMS message for the receiving system. In addition, you can set whether or not data is transferred in a transactional JMS session. Depending on your choice of the integrated EAI system (and thus the transport protocol), configuration options vary significantly and might include additional parameters.

4.6.2 SAP Business Connector (BC) Adapter

Application area

The SAP BC allows for communication with the SAP BC. As such, it is particularly useful for integration scenarios to be partially replaced with SAP NetWeaver PI.

A prerequisite for integrating SAP BC is that it must be at least version 4.7. The SAP BC adapter can process RFC and IDoc XML documents. The adapter calls, however, are stateless. This means that transactional sessions are not possible, and the EOIO Quality of Service (QoS) is not available. In addition, it cannot process attachments.

Configuration

As the sender, the adapter settings are very few, especially because the majority of these are done in SAP BC. As a sender, the SAP BC adapter provides the parameters storage period, repeat interval, repeat amount, and timeout. The message protocol setting determines if RFC or IDoc documents are received.

In the SAP BC itself, however, it is specified that the transport takes place as XML in the SAP XML dialect. In addition, the URL of the BC adapter is specified, which is structured according to the scheme *http://<pi-hostname>:<j2ee-port>/MessagingSystem/receive/BcAdapter/BC*. In addition to the parameters mentioned earlier, using the BC adapter in the receiver role requires the URL of the receiving SAP BC and the corresponding access data.

4.6.3 Plain HTTP Adapter

Using the plain HTTP adapter

The SAP NetWeaver PI plain HTTP adapter allows you to receive and send data in pure HTTP format. This is important when integrating business systems that cannot create or process SOAP documents. The receiving HTTP adapter is addressed using the URL *http://<pi-hostname>:<abap-port>/sap/xi/adapter_plain?<query-string>*. The query string contains control data (such as the sender service, namespace, and interface) which allows it to be identified and assigned to an appropriate receiver agreement. The payload itself is sent in an HTTP post as an XML document using the UTF-8 code page. Security settings, such as the use of HTTPS, can be set in the communication channel.

The plain HTTP adapter supports all QoSs. When used in synchronous mode, the HTTP adapter can return feedback about errors or success using the HTTP return code.

4.6.4 Java Proxy Generation

News in Java proxy generation

The Java proxy generation of the Enterprise Services Builder is only relevant for customers who use SAP XI 3.0, used SAP NetWeaver PI 7.0, and are planning to perform an upgrade to SAP NetWeaver PI 7.1, or have already performed such an upgrade. The current Java proxy generation in release 7.1 is designed for minor adjustments of service interfaces, which have been created due to the automatic migration of message interfaces of an Integration Repository (release XI 3.0 and SAP NetWeaver 7.0) to the Enterprise Services Repository. This migration takes place during the import of message interfaces in the Enterprise Services Repository.

Using the proxy generation of the Enterprise Services Builder, you can subsequently generate proxy objects for these service interfaces. For service interfaces that were created in release 7.1, this is not an option. The Java proxy generation is therefore no longer supported in release 7.1 or subsequent releases of SAP NetWeaver PI. For new developments, SAP recommends the Java proxy generation in the SAP NetWeaver Developer Studio.

4.6.5 RosettaNet Implementation Framework (RNIF) Adapter

Connectivity to the RosettaNet

The RNIF adapter supports the RNIF communication standard defined by RosettaNet, which specifies the protocol versions 1.1 and 2.0. The RNIF adapter for SAP NetWeaver PI is based on these versions.

The task of the RNIF adapter is to change the PI message format to RosettaNet message format. It has the ability to send messages from the PI system into a RosettaNet-compliant system, and to receive messages from a RosettaNet-compliant system.

The *SAP Business Package for RosettaNet* provides an integrated solution for enterprise-wide trading, based on high-tech industry standards. For more information, refer to the SAP Service Marketplace (*http://service.sap.com*).

4.6.6 CIDX Adapter

Standard of the chemical industry

The CIDX adapter supports the *Chem eStandards*, a data exchange standard published by the chemical industry, and is used for collaborative Internet commerce in the chemical industry. The CIDX adapter is based on the Chem eStandards encryption and security specifications, with certain exceptions derived from an extended section of the RosettaNet Implementation Framework in version 1.1.

The CIDX adapter is used for sending messages between the IS and a Chem eStandards business transaction–compatible system; the message format from the SAP NetWeaver PI is converted into a CIDX transaction message format. For more information on CIDX, see *http://www.cidx.org*.

Index