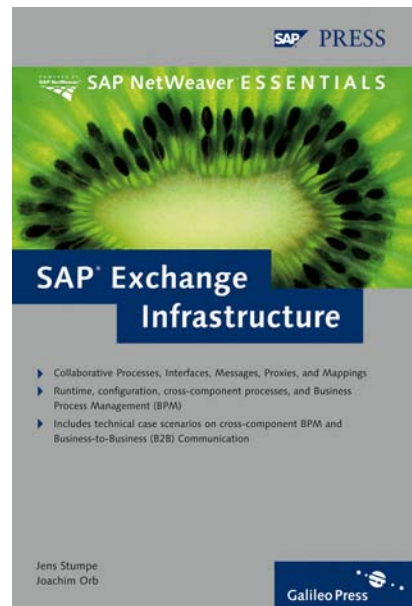


Jens Stumpe, Joachim Orb

SAP® Exchange Infrastructure



SAP PRESS

Contents

Introduction	9
1 Overview	13
1.1 Introduction	13
1.1.1 SAP NetWeaver	13
1.1.2 Levels of Process Modeling	17
1.2 Process Integration with SAP XI	19
1.2.1 Communication Using the Integration Server	19
1.2.2 Design and Configuration	25
2 First Steps	31
2.1 Overview	31
2.2 Introduction to the Integration Builder	34
2.3 Demo Examples	39
3 Designing Collaborative Processes	43
3.1 Introduction	43
3.2 Development Organization	43
3.2.1 Describing Products in the Software Catalog	44
3.2.2 Organization of Design Objects in the Integration Repository ...	46
3.2.3 Object Versioning and Transport	49
3.3 Modeling the Collaborative Process	51
3.3.1 Mapping Application Components to Systems	52
3.3.2 Modeling the Message Exchange	57
4 Interfaces, Messages, and Proxy Generation	65
4.1 Introduction	65
4.2 Developing Using the Proxy Model	66
4.2.1 Interface Development in the Integration Builder	66
4.2.2 Proxy Generation	71

4.3	Supporting Adapter-Based Communication	78
4.3.1	Importing Interfaces and Message Schemas	79
4.3.2	Developing with Imported Interface Objects	82
4.4	Enhanced Concepts	83
4.4.1	Using Message Types Across Components	83
4.4.2	Enhancing Partners' and Customers' Data Types	85
4.4.3	Accessing Message Fields by Using Context Objects	89

5 Mappings 93

5.1	Overview	93
5.1.1	Mapping Programs in SAP XI	94
5.1.2	Preconfiguration and Testing of Mapping Programs	98
5.2	Java and XSLT Mappings	101
5.2.1	Java Mappings	102
5.2.2	XSLT Mappings	105
5.3	Developing Mappings in the Integration Builder	105
5.3.1	Introduction to the Mapping Editor	106
5.3.2	Mapping Functions in Message Mappings	110
5.3.3	Developing Data Type Mappings in the Integration Builder	113

6 Configuration 115

6.1	Introduction	115
6.2	Describing Systems and Services	117
6.2.1	Settings in the System Landscape Directory	118
6.2.2	First Steps in the Integration Directory	121
6.3	Configuring Internal Company Processes	125
6.3.1	Configuration Using Integration Scenarios	125
6.3.2	Overview of Configuration Object Types	128
6.3.3	Value Mapping	135
6.4	Configuring Cross-Company Processes	137
6.4.1	From Internal to Cross-Company Communication	138
6.4.2	Partner Connectivity Kit	142
6.5	Adapter Configuration	145
6.5.1	Overview	145
6.5.2	Special Features of the RFC and IDoc Adapters	149
6.6	Adapters for Industry Standards	153
6.6.1	RosettaNet Standards	154
6.6.2	RosettaNet Support with SAP XI	155
6.7	Transports Between the Test and Productive Landscape	158

7 Runtime 161

7.1	Introduction	161
7.2	Integration Server and Integration Engine	161
7.2.1	Basics	161
7.2.2	Processing Steps of a Message	164
7.3	Proxy Runtime	169
7.3.1	Special Features for Java Proxy Communication	173
7.3.2	ABAP Proxies and Web Services	175
7.4	Monitoring	178

8 Integration Processes 185

8.1	Introduction	185
8.1.1	What Is an Integration Process?	185
8.1.2	Integration Processes and Other Processes	187
8.2	Designing an Integration Process	188
8.2.1	Data of an Integration Process	188
8.2.2	Processing Messages	190
8.2.3	Controlling the Process Flow	194
8.2.4	Time Control and Exception Handling	196
8.2.5	Importing or Exporting Integration Processes	198
8.3	Configuring Integration Processes	199
8.3.1	Overview	199
8.3.2	Configuration Using Integration Scenarios	201
8.4	Monitoring the Execution of an Integration Process	204
8.4.1	Analyzing the Runtime Cache	204
8.4.2	Process Monitoring	204
8.4.3	Message Monitoring	205

9 Cross-Component Business Process Management at the Linde Group 207

9.1	Business Background of the Scenario	207
9.2	Technical Description	209
9.2.1	Sending the Responses to the Warranty Claims	209
9.2.2	Arrival of the Messages on the Integration Server	209
9.2.3	Cross-Component Business Process Management	209
9.2.4	Message Outbound Channel	212
9.3	Implementing the Scenario at the Linde Group	213
9.3.1	System Landscape and Software Catalog	213
9.3.2	Designing the Integration Objects in the Integration Repository	214
9.3.3	Configuration in the Integration Directory	231
9.4	Summary	237

10 Cross-Company Communication Using SAP XI 239

10.1 Business Background of the Scenario 239

10.2 Technical Description 239

10.3 Implementing the Scenario 241

 10.3.1 Components of the UCCnet Scenario 241

 10.3.2 Development and Configuration Objects 242

 10.3.3 Using the Top-Down Approach to Create Integration
 Repository Objects 242

 10.3.4 Generating Integration Directory Objects Automatically 249

10.4 Summary 259

A Glossary 261

B The Authors 265

Index 267

Introduction

In SAP NetWeaver '04, SAP has brought together various technologies in one product. It includes, among others, SAP Enterprise Portal (SAP EP), SAP Mobile Infrastructure (SAP MI), SAP Business Information Warehouse (SAP BW), SAP Business Process Management (SAP BPM), SAP Exchange Infrastructure (SAP XI), and SAP Web Application Server (SAP Web AS). SAP XI focuses on cross-system process integration—the exchange of messages between applications.

SAP XI is not an adapter, but a component of SAP NetWeaver with an open architecture that enables you to integrate a wide range of SAP and non-SAP systems within and outside your company's boundaries. Given the diversity of systems installed in today's companies and the increase in cross-company communication, the need for support in this area is greater than ever before.

SAP XI has considered these requirements in its upgrade from SAP XI 2.0 to SAP XI 3.0. Due to enhancements in Business-to-Business (B2B) application support and cross-component BPM in particular, the decision regarding which release the book should be based on was an obvious one. You should note that where it is not expressly mentioned, this book applies to SAP XI 3.0 with SP4 (Feature Pack).

Release

Because this is the first book to be written on this topic, it is intended for all readers who need an introduction to guide them through their first steps with SAP XI. The first part describes the functions and most important concepts of SAP XI.

Target Group

Chapters 1 and **2** are essential for understanding all subsequent chapters. **Chapters 3** through **5** concentrate on design and development with SAP XI, independently of a specific system landscape. **Chapter 6** summarizes everything discussed in the previous chapters: It describes how you configure the cross-system process for a specific system landscape, based on the developments made at the logical level. The order in which topics are addressed reflects the chronological order of the corresponding steps in an SAP XI integration project. Logically speaking, **Chapter 7**, which deals with SAP XI runtime, could also be read concurrently with all the other chapters. **Chapter 8** completes the first part of the book with its description of cross-component BPM, which marks the transition from stateless to stateful communication.

To illustrate how SAP XI is applied in a business context, the second part of the book examines two customer scenarios that have been realized with SAP XI. We selected typical scenarios and believe that scenarios that are similar to our examples can be applied at other companies. Naturally, we did not want the scenarios to be merely examples, but to also be technically demanding, each spotlighting a specific function of SAP XI. **Chapter 9** describes how cross-component BPM is used as part of an XI scenario at the Linde Group. **Chapter 10** shows how the B2B features of SAP XI help connect a Customer Relationship Management (CRM) system to an electronic marketplace over the Internet.

Acknowledgements

This book could not have been written without the support of many people who, directly or indirectly, were involved in writing or checking the manuscript. First and foremost, we would like to thank Rachel Raw and Robert Sloan for translating the book into English so quickly and diligently. We are also indebted to the following colleagues from SAP XI development who found time to proofread sections from their specialist areas and resolve open questions: Jörg Ackermann, Frank Beunings, Andreas Dahl, Anton Deimel, Franz Forsthofer, Thea Hillenbrand, Frank Oliver Hoffmann, Christoph Hofmann, Jörg Kessler, Christoph Liebig, Michael Mühlberg, Stefan Rossmanith, Uwe Schlarb, Martin Tewes, Stefan Werner, and Manfred Zizlsperger. Finally, special thanks go to Florian Zimniak and the team at Galileo Press for their valuable support.

Jens Stumpe would like to thank the entire SAP XI Product Management Team for their involvement in the project, in particular Wolfgang Fassnacht for his organizational support; Alan Rickayzen for sharing his experience from previous book projects; Andrea Schmieden for her chapter on cross-system Business Process Management; Sindhu Gangadharan, Christine Gustav, Udo Paltzer, and Thomas Volmering for their useful contributions to the text; and everyone else who lightened Jens' workload during this time, enabling him to concentrate on the book. Last but not least, Jens would like to thank Jürgen Kreuziger, Margret Klein-Magar, and Sven Leukert for allowing him to work on the manuscript in conjunction with his other tasks at SAP.

Joachim Orb would also like to thank Matthias Allgaier, Thomas Grosser, Robert Reiz, Alan Y. Smith, and Xiaohui Wang for their support in both organizational and content issues. Thanks also go to Mr. Detlef Schulz from iWay Software for his critical review of Chapter 10, and to Dr. Klaus-Ulrich Meininger from the Linde Group for allowing us to include a scenario from his business area.

Joachim Orb would also like to thank Agnès Bouillé for her cooperation. Many evenings and weekends were sacrificed to produce this book.

The challenges of cross-system and cross-company processes arise from their multifaceted nature regarding the diversity of platforms, programming languages, involved applications, and communication parties. We hope that this book will give you the necessary guidance when using SAP XI to meet these challenges, and that it will contribute to the success of your integration projects.

Walldorf, February 2005

Jens Stumpe

Joachim Orb

6 Configuration

The design of the collaborative process is independent of the technical details resulting from the system landscape of the customer. This chapter describes how to configure this information centrally to control message processing at runtime.

6.1 Introduction

At design time, we look at collaborative processes at the logical level. In this view, messages are exchanged between application components and not between systems. In this chapter, we make the link between this abstraction and the settings that are required at runtime to actually implement message exchange. These settings concern the following areas:

- ▶ Information about the actual system landscape and the products installed there. This is discussed at the beginning of Section 6.2.
- ▶ Information regarding the services provided within a system landscape and which technical communication channel other systems in the system landscape use to access a service. This is discussed at the end of Section 6.2, once the basics have been covered at the beginning of the section.
- ▶ Information about how the services are linked to one another by messages (logical routing) and whether a mapping is necessary. This is described in relation to internal company communication in Section 6.3.
- ▶ Information about services that you want to make available to business partners outside your own system landscape. This is described in Section 6.4, which builds on the concepts introduced in Section 6.3.

With the exception of the area listed in the first bullet, you configure all the necessary information centrally in the Integration Directory. You have the following options:

- ▶ If there is an integration scenario for your collaborative process in the Integration Repository (see Section 3.3), we recommend that you use this scenario for configuration. This is discussed in more detail in Section 6.3.1.

- ▶ If there is no integration scenario, a configuration wizard is available to guide you through the individual configuration steps.
- ▶ Alternatively, you can make the configuration settings manually. Unlike the first option—where the Integration Builder automatically recognizes from the integration scenario which configuration objects can be reused and which objects must be generated—manual configuration is very time-consuming.

Processes and Scenarios

To avoid confusion, we want to reiterate that the term *collaborative process* means a process that exists in the real world and a process that you want to implement using your software technology. An *integration scenario* is a design object in the Integration Repository that you use to model the collaborative process. *Integration processes* are also design objects that enable you to consider dependencies between messages in cross-system message exchange.

During the configuration of a collaborative process, you can choose whether or not you want to work with an integration scenario from the Integration Repository. On the one hand, making the integration scenario an integral part of the Integration Repository would be too restrictive. On the other hand, the lack of an integration scenario means that there is nothing to hold together the configuration objects of a scenario. By way of a compromise, the Integration Builder works with *configuration scenarios*.¹ These scenarios are simply a container for all the configuration objects that are required to configure a collaborative process. Figure 6.1 shows the configuration scenario `MyCheckFlightSeatAvailability` in the Integration Builder. The **Configuration Scenario Objects** tab page shows all the objects that are assigned to a configuration scenario. In this case, the objects were generated or suitable existing objects were automatically assigned using the `CheckFlightSeatAvailability` integration scenario. However, you don't have to use an integration scenario from the Integration Repository and can assign any configuration objects of your choice to a configuration scenario.

No matter which configuration option you choose, it is critical that you understand the various configuration objects. We'll take a step-by-step look at how these objects are used below.

¹ As of SAP XI 3.0 SP9, **scenario** is renamed **configuration scenario**. We use the new terminology in this book.

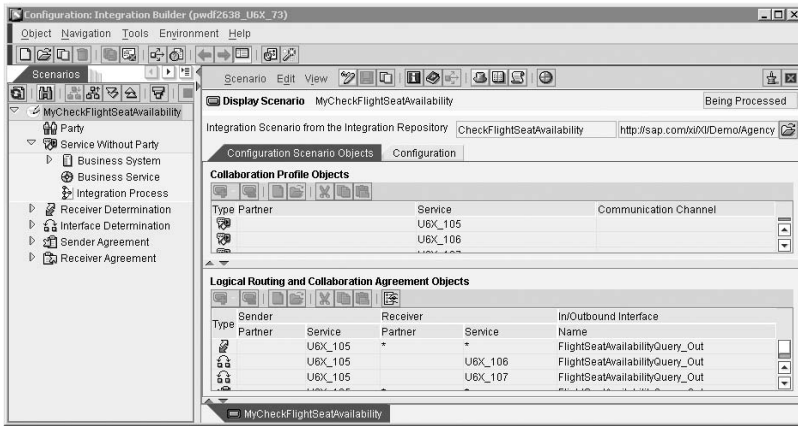


Figure 6.1 Configuration Scenario MyCheckFlightSeatAvailability

Configuration in the Integration Directory is designed to support as many configuration scenarios as possible. Depending on the protocol, the configuration scenarios can have very different technical requirements. However, the procedure for configuring the configuration objects is valid for many configuration scenarios. Sections 6.2, 6.3, and 6.4 explain this procedure and highlight any exceptions. Sections 6.5 and 6.6 address the special features of the various adapters. Finally, Section 6.7 focuses on the transport of configuration objects.

The Integration Builder also enables you to publish a service as a Web service at configuration time. Since we'll meet Web services again in a different context later in the book, we'll review this topic in its entirety in Section 7.3.2.

6.2 Describing Systems and Services

To configure a collaborative process within your system landscape, you must first describe the system landscape in the System Landscape Directory (SLD). This is discussed in more detail in Section 6.2.1. Basically, you have two options when it comes to dealing with the sequence of the configuration steps in the Integration Directory: You can work from the collaborative process (logical level) to the technical systems (technical level), or vice versa. The advantage of the latter option is that a system generally offers services for a wide range of collaborative processes and plays a role in different configuration scenarios. The technical options provided by the systems for message exchange are more constant and you only need to enter them once for all configuration scenarios in a *collaboration pro-*

file. For this reason, it is advisable to focus on this profile first, and then move on to configuration at the logical level. Therefore, we'll look at the settings in the SLD first and then examine the configuration of the collaboration profile in Section 6.2.2.

6.2.1 Settings in the System Landscape Directory

Like the Integration Builder, you call the SLD from the SAP XI start page (see Section 2.1). Besides the software catalog, which we met in Section 3.2.1, you can also enter and call the following information about your system landscape:

► Technical Landscape

In this area, you can access information about the technical systems in your system landscape. Examples of technical systems are an SAP Web AS (ABAP) or an SAP Web AS (Java).

► Business Landscape

In this area, you can access information about the business systems of your system landscape. This area is specific to SAP XI and enables you to identify those systems in your system landscape that use SAP XI to exchange messages.

Technical Systems The information about the technical systems of your system landscape is not just of interest to SAP XI users. It can also be used by SAP support employees and customers to get an overview of the installed systems:

► Technical SAP systems

The SLD categorizes the technical SAP systems by the Basis or SAP Web AS release that they run on. The following systems register themselves automatically in the SLD when they're installed: SAP Basis 4.0B, and all SAP Web AS ABAP systems and SAP Web AS Java systems as of Release 6.40. They also transfer data about their installed products. You must register all other technical SAP systems in the SLD manually by using a wizard, and then assign them products from the software catalog.

► Third-party systems

You also register third-party systems in the SLD manually by using a wizard. You can assign these systems third-party products from the software catalog.

The technical attributes of a system are stored in the SLD. Examples of attributes for technical SAP systems are: system name, system clients, message server, and installed products. Furthermore, you can use the

Exchange Infrastructure option to display all technical SAP systems on which SAP XI runtime components are installed, for example, the Integration Server. These components register themselves automatically in the SLD as soon as they're launched.

If a technical system is part of a cross-system process, you must also assign it to a business system. (In SAP systems, every client represents a business system.) During configuration, you then work with the name of the business system and not with the name of the technical system. First, this ensures that only those systems relevant to the process are displayed during configuration. Secondly, you can make changes to the technical system landscape without affecting an existing configuration.

Figure 6.2 Attributes of a Business System in the SLD

Business systems are used exclusively for cross-system applications with SAP XI. Therefore, the attributes of a business system in the SLD relate directly to the particular application case. Figure 6.2 shows a screenshot of a business system in the SLD. As well as the header data (**Name**, **Description**, **Administrative Contact**), you must also define the role of the business system.

- If it is an application system, you must assign it an Integration Server, with which the business system will exchange messages. Since you usually test the message exchange before using the process in a productive environment, there can be multiple Integration Servers within a system landscape.

- ▶ Alternatively, you assign the business system the role of an Integration Server.

Just like the other data in the SLD, this information is merely descriptive. Therefore, defining a business system as an Integration Server in the SLD does not relieve you of the task of making the corresponding configuration settings for the respective clients in the technical system (see Section 7.2.1). Other SAP XI runtime components may also call data in the SLD. As will be discussed again later in Section 6.7, the **Group** and **Transport Targets** attributes are required for the transport of configuration objects between different Integration Directories.

Evaluating the SLD Data

We will now focus on the uses of the data in the SLD for configuration in the Integration Directory. At the bottom of the screenshot in Figure 6.2, you can see the first of the installed products (SAP EXCHANGE INFRA-STRUCTURE). The products listed here and the derived software component versions are used by the assigned technical system (in this case, the client 106 of SAP system U6D). Since business systems are used in the Integration Directory to configure internal company communication, the Integration Directory accesses information about business systems and associated technical systems from the SLD to derive further details. For example, the Integration Builder can use the software component versions of a system to determine all the interfaces for a business system that have been saved in the Integration Repository for message exchange. Figure 6. illustrates this query. The Integration Builder also uses this mechanism for checks and input help.

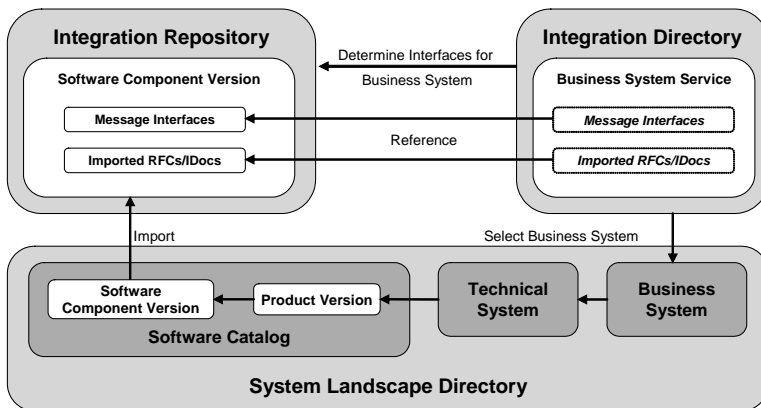


Figure 6.3 Referencing Content of the Integration Repository

The use of the software catalog in the Integration Repository, discussed in Section 3.2.1, completes the loop. The *business system service* shown in Figure 6.3 leads us to our next topic: the Integration Directory. The next section explains the configuration procedure in the Integration Directory.

6.2.2 First Steps in the Integration Directory

The Integration Builder provides a whole range of objects for configuring a collaborative process. Before we delve into how to use these configuration objects in detail, let us first get a general overview. Because there are dependencies between the various objects, it is advisable to adhere to the following sequence during configuration:

1. The configuration objects *communication party*, *service*, and *communication channel* reference each other and together form a *collaboration profile*. To exchange internal company messages, it is generally sufficient to use collaboration profiles, where the services and communication channels are specified. Therefore, for now, we'll leave the discussion of communication parties, and return to it in Section 6.4.
2. At this stage, the collaboration profile is still independent of a specific configuration scenario. You use the configuration objects *sender agreement* and *receiver agreement* to define the communication options that you want or have to use. These agreements are collectively referred to as a *collaboration agreement*.
3. Finally, you use *receiver determinations* and *interface determinations* to configure the logical routing, which defines where a message should be forwarded and whether a mapping is necessary beforehand.

The description of the collaboration profile is the basis for the following configuration steps. You can use the profile in different configuration scenarios. Therefore, let's take a closer look at this area before moving on to the configuration of internal company processes in Section 6.3.

Figure 6.4 shows the object hierarchy of communication parties, services, and communication channels. As we already mentioned, we'll look at communication parties in more detail later on. At this point, you need only know that a company uses a communication party to enter the services provided by a business partner in the Integration Directory. Therefore, you don't actually need a communication party as a configuration object for internal company processes. (The exception to this is a configuration scenario with IDocs. We'll look at this in more detail in Section 6.5.2.)

Collaboration Profile

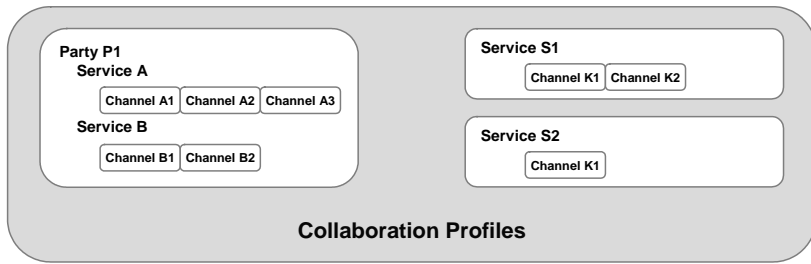


Figure 6.4 Object Hierarchy in Communication Profiles

Service The *Service* object was introduced with SAP XI 3.0 as an additional level for business systems to enable other sender and receiver types to be modeled and addressed. There are three types of services:

► **Business system service**

This service refers directly to a business system from the SLD. To create business system services, you call the context menu for the **Service Without Party** or **Business System** node in the Integration Builder navigation tree and choose **Assign Business System...**

► **Integration process service**

This service refers to integration processes from the Integration Repository. Configuration with integration processes is discussed in more detail in Chapter 8.

► **Business service**

This service enables business partners to address receivers of your system landscape without you having to publish the receivers. We will look at this in more detail in Section 6.4.

A service offers a range of interfaces for communication using SAP XI. These interfaces are displayed on the Sender and Receiver tab pages. In the previous section, Figure 6.3 showed that the Integration Builder automatically determines these interfaces for business system services from the Integration Repository. Consequently, they're displayed in the input help (that is, the help that users can call up to enter a value in an entry field) in subsequent configuration steps. If the interfaces of a business system are not in the Integration Repository because they have not been created or imported, you must enter them manually in the subsequent configuration steps.

Communication Channel

When creating a business system service, the Integration Builder automatically creates communication channels for the service, which you must then adapt to your configuration scenario:

- ▶ For an SAP system, separate receiver channels are generated for RFC, IDoc, HTTP, and proxy communication (**Adapter Type XI**).
- ▶ For a non-SAP system, an HTTP receiver channel is generated.

Communication channels define the inbound and outbound processing in the Integration Server. To start with, the channels of a business system service simply reflect the options in the business system for receiving and sending messages. You define the channel to be used for a selected communication for the sender or receiver by using the collaboration agreement.

You may be asking yourself which system a *receiver* channel refers to: the Integration Server or the application system? This is a good question, and Figure 6.5 provides the answer. The communication channel for the sender configures the sender adapter, which converts the sender message for more processing in the Integration Server. Therefore, the channel for the sender determines the inbound processing in the Integration Server. Outbound processing works in a similar way. All configuration object names in the Integration Directory are based on the symmetry displayed in Figure 6.5. A configuration object for the receiver always refers to the receiver application system or the receiver business partner, and *not* the Integration Server sending the message.

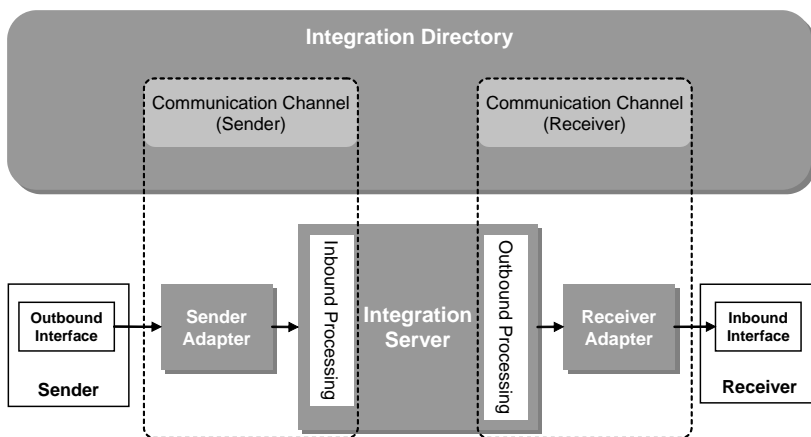


Figure 6.5 Sender and Receiver in Configuration

The representation of the adapters in Figure 6.5 reflects the logical point of view. It does not show where the runtime components of the adapters are actually installed. The proxy runtime, in particular, is installed in the application system. Nevertheless, you configure the proxy runtime in the

Sender and Receiver Configuration

same way as the other adapters, that is, by using a communication channel and choosing **Adapter Type XI**. We will look at adapter configuration in more detail in Section 6.5. You should now understand the symmetry of the configuration objects with respect to the sender and receiver. In configuration, however, you don't always require both sides. Since the Integration Server must define a receiver for the message, the configuration objects for the receiver side are mandatory. On the sender side, however, whether you have to configure anything depends on the adapter type and the configuration scenario. For example, the proxy runtime in the sender application system uses information from the SLD to determine the address of the Integration Server. Therefore, in this case you make configuration settings in the Integration Directory on the sender side only if security settings are required for message transfer.

Let's take another look at the object hierarchy of the collaboration profile from Figure 6.4. In the Integration Repository, we saw that namespaces ensure that object names are unique. Objects in the Integration Directory don't have any namespaces. Instead, the name of the higher-level object type serves as the namespace in collaboration profiles. It is normal for two communication channels of different services to have the same name, since the adapter type of the channel is often the same. For example, the names of the communication channels generated for business system services are always the same. The configuration of the channels, on the other hand, is specific to the business system.

Communication Channel Templates

For those adapter types where no communication channels can be generated, it would prove laborious to always have to edit the frequently used attributes manually. Certain attributes are often known at design time. For example, the *RosettaNet* industry standard stipulates security settings (encryption, signature) for the respective *Partner Interface Processes* (PIPs). To accelerate the configuration of such scenarios, the Integration Builder provides communication channel templates, which you create in the Integration Repository and reuse in the Integration Directory. Section 6.6 explains how SAP XI supports industry standards. You can use communication channel templates for all adapter types.

6.3 Configuring Internal Company Processes

So far, we have looked at the basic settings that form the foundation for a whole range of configuration scenarios. This section focuses on internal company scenarios. In Section 6.3.1, we use a demo example² to explain the configuration, using the corresponding integration scenario. In this case, the Integration Builder supports the automatic generation of configuration objects using information from the integration scenario. We will use this example to explain the general concepts in the following sections.

6.3.1 Configuration Using Integration Scenarios

We were introduced to the integration scenario `CheckFlightSeat-Availability` in Section 3.3. This scenario models a flight availability check in which a travel agency exchanges messages with one or more airlines. In the demo example, there are two airlines. For the purpose of simplification, let's say that these airlines are two different clients of the same SAP system:

- ▶ Client 105 is the travel agency on the sender side.
- ▶ Clients 106 and 107 are the airlines Lufthansa and American Airlines, respectively, on the receiver side.

The following steps are based on the assumption that the description of the technical systems and the business systems is already contained in the SLD. When writing this book, we worked with SAP system `U6X` and created the business systems `U6X_105`, `U6X_106`, and `U6X_107`. We also created the corresponding business system services, as described in Section 6.2.2, and generated and adapted the required communication channels.

Now we'll explain how to configure this scenario in the Integration Builder. To do this, first choose **Tools • Transfer Integration Scenario from Integration Repository...** in the main menu to access the integration scenario from the Integration Repository. This creates a configuration scenario that references our integration scenario. After the transfer, the **Integration Scenario Configurator** dialog box appears, which displays the first **component view** of the integration scenario (see Figure 6.6).

² The demo example is described in Section 2.3.

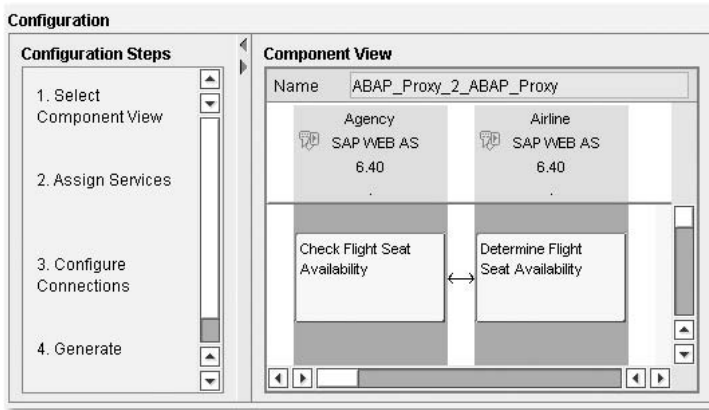


Figure 6.6 Configuration Steps for Integration Scenarios

To make the configuration settings, perform the configuration steps in the order displayed on the left side of the figure. To do this, click on the respective step and assign the required configuration objects in the dialog box that appears:

1. In the first configuration step, specify the component view (see Section 3.3.1). In this case, we keep the component view that is already selected, ABAP_Proxy_2_ABAP_Proxy.
2. In the second configuration step, assign a service to each application component. The dialog box that appears displays the first application component Agency. Assign the business system service U6X_105 to it. Use the blue navigation arrow to switch to another application component Airline. Assign the services U6X_106 and U6X_107 to it.
3. The third configuration step deals with connections (see Figure 6.7). Each receiver service requires a communication channel to enable the Integration Server to forward the respective message to the technical system. On the sender side, on the other hand, it is not necessary to configure a communication channel for the XI adapter (this is discussed in detail in Section 6.5).

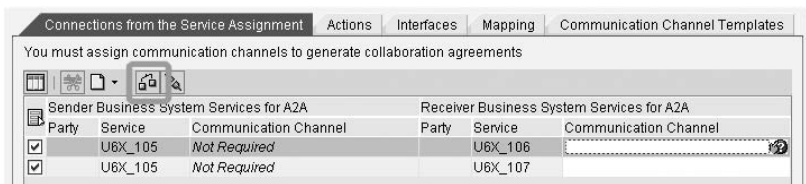


Figure 6.7 Configuring Connections

Let's concentrate for the moment on how to select a communication channel. In Section 6.2.2, we learned that a *collaboration agreement* is required to select a particular communication channel at runtime. This example deals with a *receiver agreement* for the message to the *receiver* business system U6X_106 or U6X_107. There are two different cases:

- ▶ If there is no existing receiver agreement that matches the receiver business system and the inbound interface, you must assign the required communication channel using the input help. An appropriate receiver agreement is generated later. Figure 6.8 shows the dialog box where you make this selection. All communication channels that are available for the receiver service are displayed.
- ▶ If a receiver agreement already exists for your receiver, you can use the function circled in red in Figure 6.7 to automatically define the channel. In this case, the dialog box displays only the communication channel defined by the receiver agreement in the communication channel selection. If the receiver agreement is for all inbound interfaces of the receiver system (referred to as a *generic* receiver determination), you can expand the communication channel selection by creating a *more specific* receiver determination, that is, one that is intended for a specific inbound interface. Section 6.3.2 addresses generic and specific configuration objects.

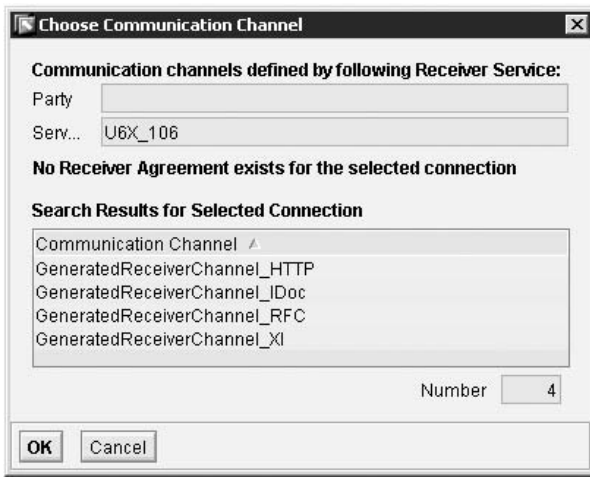


Figure 6.8 Selecting a Communication Channel

4. Finally, once you have made these preparations, you can have the integration scenario configurator generate all the remaining configuration

objects. You can restrict this generation to particular object types. To check the generation without creating new configuration objects, you can also simulate the procedure. In both cases, the Integration Builder shows the results in a detailed generation log. Figure 6.9 shows a screenshot of the log. The traffic lights in the log represent generation steps with errors (red traffic light), incomplete generation steps (yellow traffic light), and complete generation steps (green traffic light). If a generation step is incomplete, this means that you may have to add information that cannot be generated automatically, for example, routing conditions.

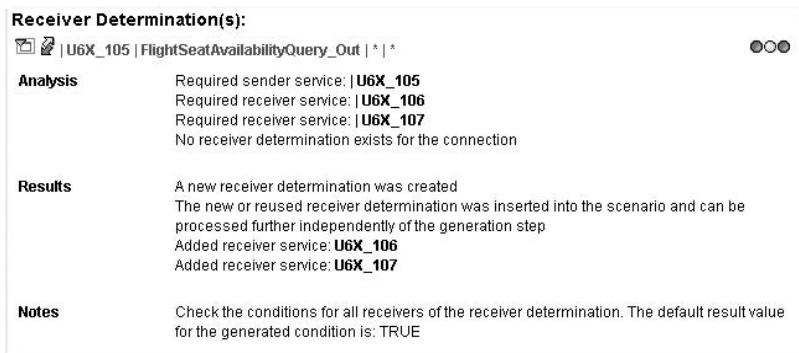


Figure 6.9 Generation Log

The Integration Builder automatically adds all generated and reused objects to the object list of your configuration scenario. You can also use the context menu for the connection to display the configuration objects belonging to each connection in the component view.

To finish the configuration, work through the generation log by navigating directly from the log to the corresponding objects and adding the missing information. The next section deals with the background knowledge necessary to complete the configuration.

6.3.2 Overview of Configuration Object Types

We've already looked at the configuration objects of the collaboration profile: party, service, and communication channel. Before examining other configuration objects, let's see how all the objects are related to one another.

Key Fields

Unlike the objects in the Integration Repository, the objects here are not organized using software component versions. Therefore, all configuration objects become globally visible in the Integration Directory as soon as they are released and are simultaneously activated for the runtime environment. Consequently, it's worth taking a closer look at the key fields of the objects. To simplify this overview, the objects are separated into three tables according to their use.

Key Fields	Object Type		
	Party	Service	Communication Channel
Service	(X)	X	
Communication Channel	(X)	X	X

Table 6.1 Key Fields for Objects of the Collaboration Profile

There is little more to say about the key fields of the objects of the collaboration profile in Table 6.1. There are services without a party, but no parties without a service. In objects without a party, the key field remains initial. The communication channel consists of the key fields of the higher-level service and its own name. As we have already determined, you choose the objects of the collaboration profile during the remaining configuration steps and determine the relationships between them. Therefore, the key fields of the service are always part of the key of the other configuration objects.

Key Fields	Object Type	
	Sender Agreement	Receiver Agreement
Sender party	(X)	(X)* (Header mapping)
Sender service	X	X* (Header mapping)
Outbound interface	X	
Namespace of the outbound interface	X	
Receiver party	(X)*	(X) (Header mapping)

Table 6.2 Key Fields for Sender and Receiver Agreements

Key Fields	Object Type	
	Sender Agreement	Receiver Agreement
Receiver service	X*	X (Header mapping)
Inbound interface		X*
Namespace of the inbound interface		X*

Table 6.2 Key Fields for Sender and Receiver Agreements (cont.)

Table 6.2 focuses on collaboration agreements. To ensure that the information in the table is complete, we added the following additional information to the table:

- ▶ The values from four fields of the receiver agreement can be mapped to other values using a header mapping. We'll examine the reasons for doing this in the cross-company scenarios in Section 6.4.
- ▶ Key fields marked with an asterisk (*) can be filled *generically*. Don't confuse these fields with the input fields in the Integration Builder that are marked with a red asterisk. The latter are *required* fields.

Generic and Specific Fields

You use generic fields to define the configuration for multiple cases by entering an asterisk in the field. For example, you can create a receiver agreement independently of a specific inbound interface. During message processing, the Integration Server checks for receiver determinations with matching key fields and selects the most specific. In some constellations, the Integration Builder cannot determine this due to overlapping. The Integration Builder checks this and notifies you during creation if this is the case. You must also be aware that generic configurations are valid globally in the Integration Directory. If several configuration scenarios use the same generic object, any changes to this object will result in side effects for all these scenarios.

The remaining two configuration objects are for logical routing. The key fields are displayed in Table 6.3. The virtual receiver is relevant to only cross-company communication, which we will look at in Section 6.4. If you don't specify a virtual receiver when creating a receiver determination, the Integration Builder inserts an asterisk for both fields (in other words, the receiver determination is independent of a virtual receiver).

We will now expand on this brief overview and look at the individual object types and their uses in more detail.

Key Fields	Object Type	
	Receiver Determination	Interface Determination
Sender party	(X)*	(X)*
Sender service	X*	X*
Outbound interface	X	X
Namespace of the out-bound interface	X	X
Receiver party	(X)* <i>(Virtual receiver)</i>	(X)*
Receiver service	X* <i>(Virtual receiver)</i>	X*

Table 6.3 Key Fields for Objects of Logical Routing

Collaboration Agreements

Senders and receivers of a message use a collaboration agreement to agree on the communication channel to be used to exchange messages. The obvious question here is what is meant by *sender* and *receiver*, since the Integration Server sends and receives messages, as do the application systems. Logically speaking, the Integration Server is situated between the application systems, therefore we need not just *one*, but *two* communication channels: one between each application system and the Integration Server. For this reason, there are collaboration agreements that define the channel on the sender side, and those that define the channel on the receiver side. Figure 6.10 illustrates this symmetry and the corresponding sender and receiver agreements.

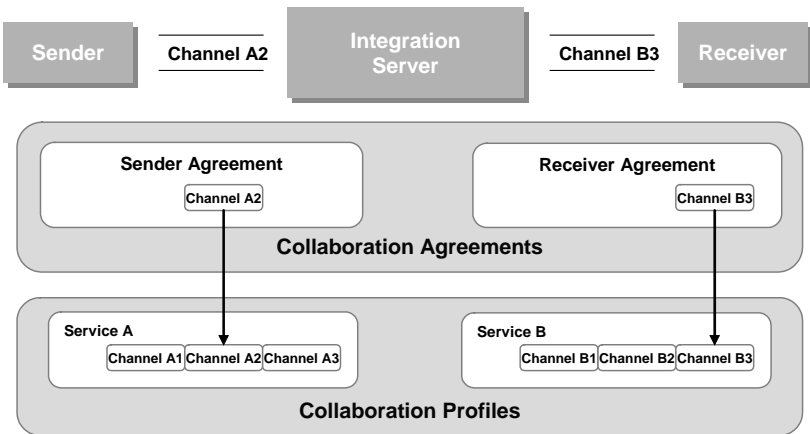


Figure 6.10 Sender and Receiver Agreements

You can see from the key fields of collaboration agreements in Table 6.2 that both the sender agreement and the receiver agreement have the sender service and the receiver service in their key. They are always intended for a communication pair, but each configures just one side of the communication with the Integration Server.

Sender and Receiver Agreement

Receiver agreements are *obligatory*, since the Integration Server must know which adapter to forward the message to. The situation is different on the sender side, because the sender adapter can use information from the SLD to determine the address of the Integration Server. In Section 6.5, we examine in more detail when sender agreements are necessary and why. The communication channel for the sender is also not absolutely necessary if the adapter can find the required configuration data itself.

Security Settings

The RNIF, CIDX, XI, and marketplace adapters also support security settings (signatures, authentication). The corresponding attributes are part of the respective communication channel, where you define whether and which security settings are supported. You configure these settings for a specific connection in the collaboration agreement.

Receiver and Interface Determination

The remaining task is to configure the logical routing. Logical routing has two steps:

- ▶ You use a *receiver determination* to define one or more receiver services for a message. You can define a condition for each receiver service in XPath or with context objects (see Section 4.4.3). There is no guaranteed receiver sequence for receiver services, and this is not important in stateless message processing. You encounter such requirements using integration processes, which are discussed in Chapter 8.
- ▶ You use an *interface determination* to define one or more inbound interfaces as receiver interfaces for the message. In this case, you define a receiver sequence using the sequence in which you enter the inbound interfaces in the interface determination.

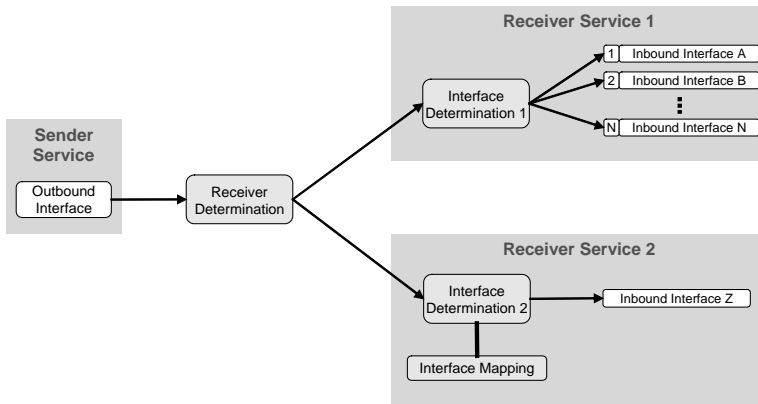


Figure 6.11 Example of Logical Routing

In the example in Figure 6.11, there are two receiver services, which are configured using a receiver determination for a sender service and an outbound interface. If conditions are specified in the receiver determination for the forwarding of the message, it is not a problem if these overlap. The Integration Server copies³ the message for each true condition, generating a new message ID for each receiver service.

Copying Messages

Whether an interface determination is required depends on the configuration scenario. If a mapping is necessary, you definitely need an interface determination to configure the selection of mapping programs. We saw in Section 5.1.2 that you can bundle mapping programs for an interface pair by using an interface mapping. If you need to use a mapping, specify the interface mapping in the interface determination. If you use an integration scenario to generate the configuration, the interface mapping is entered automatically.

Configuring Mapping Programs

A mapping is not always necessary, because the sender and receiver both use the same interface technology, for example IDoc-IDoc communication or RFC-RFC communication using the Integration Server. In these cases, the name and namespace of the interface remain the same throughout the entire message transfer, and it is therefore not necessary to determine an interface. Outbound and inbound *message* interfaces, however, are located in different namespaces or have different names, which means that an interface determination is always necessary (even if no mapping is needed).

³ Somewhat misleadingly, this process is sometimes referred to as a *message split*, even though the message cannot be divided into smaller messages at this point. To split or merge messages, you need an integration process.

Configuration Overview

Besides the configuration scenarios, which bundle all the configuration objects of a scenario together, the Integration Builder also provides a *configuration overview*. This overview focuses on all objects that are required to process and forward messages to the receiver once the inbound processing in the Integration Server is complete: the receiver determination, the interface determination, and the receiver agreement. The Integration Server first uses the sender information in the message header to determine the configured receiver or receivers (receiver determination), then the configured inbound interface or interfaces and a corresponding interface mapping (interface determination), and finally, the communication channel (receiver agreement). Figure 6.12 illustrates this relationship. Information that arrives after the receiver determination is shown in gray.

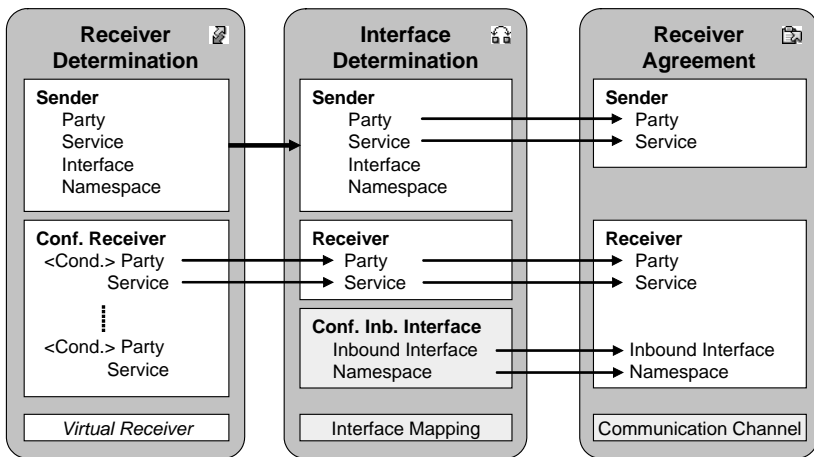


Figure 6.12 Configuration Based on Receiver Determinations

Since the processing steps proceed from the receiver determination, this is where the configuration overview is located. You can add configuration information to this table and navigate directly to the configuration objects listed there by double-clicking them.

The concepts that we have introduced so far are essentially sufficient to configure internal company configuration scenarios. However, even within a company it may be the case that the sender and receiver identify the same object in different ways. If this is the case, the receiver would then interpret the corresponding values in the message payload incorrectly. The next section explains how to handle such ambiguity using SAP XI.

6.3.3 Value Mapping

If the same objects are identified differently at the sender and receiver sides, you need a value mapping. As we saw in Section 5.1.1, SAP XI has a value-mapping table for all value mappings. Before we explain how to enter the source and target values of an object in the table, let's take a closer look at a table entry in Figure 6.13. To illustrate this point, we have taken a fictitious table entry from the Business-to-Business (B2B) world. This example involves a long-serving SAP employee, who orders from an online music store (Thomann) in his free time. At SAP, he is identified uniquely by his *employee number* (D000002). The online music store is not aware of any employee numbers and instead identifies the same person with a *customer number* (05940). Though unlikely, if SAP were to offer a service whereby its employees could place orders with the online music store using a B2B application and have the payments deducted from their salaries, these values would have to be mapped to each other.

Agency	Identification Scheme	Value	Agency	Identification Scheme	Value
SAP	EmployeeId	D000002	Thomann	CustomerId	05940
...




Figure 6.13 Entries in the Value-Mapping Table

A person is identified differently at SAP than at Thomann. We talk about different *representations* of the same object. The important thing is that the person in our example can be identified by the following trio:

- ▶ An *issuing agency*, which defines how an object (in our example a person) is to be uniquely identified. In our example, the issuing agency is the company SAP or the online music store.
- ▶ The agency uses an *identification scheme* to identify the object. In our example, this is the employee number or the customer number.
- ▶ The actual *value* for identification, according to the conventions of the identification scheme.

You use this trio (agency, identification scheme, value) to identify the *representation* of an object. It is up to you which representations you use in the value-mapping table, and depends on how the value of the representation is defined. The above example addresses cross-company commu-

Identifying Representations

nication. In the case of internal company communication, the following cases are possible:

- ▶ The issuing agency of the representation can be determined by the application components that exchange messages with each other, for instance APO or CRM.
- ▶ In a productive landscape, you can identify objects by their technical unit. In this case, the agency is the name of the business system and the identification scheme is determined by the object type (for example, the `Customer` object type for a business object).

If you want to execute value mappings within a Java mapping or a message mapping, you reference the values to be mapped by specifying the agency and the identification scheme in the mapping. Therefore, you need to consider how to identify values that are to be mapped at design time. You can enter the values in the value-mapping table in the following ways:

- ▶ Use the Integration Builder (Configuration) to enter all representations of an object in a **Value-Mapping Group**. Create a value-mapping group and enter all representations of the same object. To display the resulting value mappings, choose **Tools • Value Mapping...** in the main menu.
- ▶ Use the message interface `ValueMappingReplication` that is defined in the software component version `SAP Basis 6.40` in the namespace `http://sap.com/xi/XI/System`. SAP ships a Java server proxy for this inbound message interface, which executes the mass filling of the value-mapping table. For this filling, you implement the outbound side and configure the communication in exactly the same way as for any other configuration. Choose adapter type **XI** for your communication channel and specify the Java proxy runtime that runs on the same SAP Web AS on which the Integration Server is running.

The advantage of the first method is that it has object versioning and a transport connection for value-mapping groups. In the latter case, you access the value-mapping table in the runtime cache of the Integration Server directly. Therefore, you cannot call and edit entries made in this way in the Integration Builder.

The agency and identification scheme are not only relevant to value mappings, but also to cross-company communication, because there can be different representations for communication parties in B2B applications as well. This is discussed in more detail in the next section.

Index

A

- ABAP proxy generation 73
 - converting names 75
 - metadata 75
- ABAP proxy objects 76
- Acknowledgment 165, 171
- Acknowledgment message 68
- Action 59
- Actions 57, 245
- Adapter 78
 - JDBC adapter 79
- Adapter Engine 20
- Adapter Framework 22
- Adapter metadata 145, 159
- Adapter-specific identifier 149
- Agency 135
- Aggregation 228
- ALE scenario
 - Linde 209
- Alert Configuration 182
- Application component 53, 243
- Application error 172
- Authentication 147

B

- B2B communication 239
- Bean classes 173
- Best Effort 164
- Business process → Integration process
- Business Process Engine 20, 186, 199
- Business Process Management 15, 187, 207, 209
- Business scenario → Integration scenario
- Business service 140
- Business system 54, 119, 251
- Business system groups 159

C

- Caches 39
- ccBPM 187, 207
- CCMS 179
- Change list 35, 49
- Client 150

- Collaboration profile 118, 129
- Collaborative process 19, 116
- Collect pattern 209
- Collecting IDoc contents 224
- Commit handling 167
- Communication channel 127, 233, 252
- Communication channel template 124
- Communication party 58, 139, 249
- Component view 56, 255
- Configuration
 - Collaborative processes 32, 115
 - Technical 31
- Configuration objects 26, 242
 - Key fields 129
- Configuration overview 134
- Configuration scenario 116, 231
- Configuration time 26
- Container elements
 - for alerts 183
 - In integration process 189
- Container operation 211
- Context object 90
 - For integration processes 201
 - technical 90
- Copy function 84
- Correlation 210
 - activating 221

D

- Data type 68, 69
 - built-in data type 71
 - complex type 71
 - enhancement 85
 - repository namespace 70
 - simple type 71
- Data type mappings 95, 113
- Data-flow editor 110, 111
- Denormalization 139, 141
- Design objects 26
- Design time 26
- Development object 242
- Document object model 102

E

- EJB → Enterprise Java Beans
- Endless loop 223
- Enterprise Java Beans 173
- Exactly Once 164
- Exactly Once In Order 164
- Exception Handling
 - In Integration Processes 197
- Export function
 - In interface mappings 102
 - of BPEL 64
 - of WSDL 68
- External definition 81

F

- Fault message 68
- ForEach 211
- Fork 210

G

- Generation log 128
- Global configuration data 162

H

- Hoplist 165
- HTTPS 147

I

- IBM WebSphere message queue 252
- Identification scheme 135, 138
- Identifier 138
- IDoc adapter 149
- IDoc import 216
- IDoc partner 150
- IDoc-IDoc communication 207
- IDoc-XML 149
- Imported Archive 248
- Importing interfaces 214
- Inside-out 79
- Integration Builder 26, 34
- Integration Directory 26
 - Linde 231
- Integration Engine 20, 162, 199
 - Local 20, 170
 - pipeline 163
- Integration object
 - design 214
- Integration process 55, 96, 116, 185

- Block 188
- Configuration 199
- Container elements 189
- Correlation 192
- Deadline monitoring 197
- Dynamic processing 195
- Export 198
- Fork 194
- Local correlation 196
- Loop 194
- Predefined 186
- Receive step 190
- Runtime cache 204
- Send step 191
- signature 218
- Step types 188
- Switch 194
- Transformation step 192
- Integration Repository 26, 214
- Integration Repository objects
 - Creating 242
- Integration scenario 51, 57, 116, 243
 - References 59
- Integration scenario configurator 240, 254
- Integration Server 162
- Integration-Directory objects
 - Generating 249
- Interface description 65, 79
- Interface determination 98, 132, 233
- Interface mapping 97, 98, 248
- Interface Repository 79
- Issuing agency 138

J

- Java mapping 104
- Java proxy generation 77
 - naming conflicts 77
 - regeneration 77
- Java™ Web Start 34
- JMS adapter 241, 252
- JPR Registry 174

L

- Linde 207
- Logging 168
- Logical routing 233
- Logical system 150

M

- Mapping editor 97, 106
 - Context 110
 - Data-flow editor 106
 - Overview functions 108
 - Queues 110
 - Standard functions 111
 - Structure overview 107
 - Target structure 109
 - Target-field mappings 106
 - User-defined function 112
- Mapping program 94, 143
 - In interface mappings 99
 - In Java 104
 - In XSLT 105
- Mapping template 113
- Mapping trace in Java programs 104
- Mappings
 - Imported archives 102
 - Test environment 100
 - Trace level 100
- Message header 89
- Message ID 166, 167, 171
- Message instance 83, 88
- Message interface 66, 67, 174, 246
 - Abstract 99, 158, 190
 - abstract 67
- Message mapping 105
 - Complete 109
- Message monitoring 181
- Message outbound channel 212
- Message processing
 - asynchronous 165
 - prioritized 166
 - synchronous 167
- Message schema 81, 82
- Message type 68
 - fault message type 68
 - input message type 68
 - output message type 68
- Messages
 - collect 209
 - definition 216
 - merge 211
 - send 211
 - sort 210
- Messaging system 170
- Monitoring

- alerts 182
- end-to-end 181
- of caches 179
- of components 179
- Of integration processes 204
- Of messages 205
- performance 179, 182

Multi-mappings 96, 97, 102, 109

N

- Namespace 215
 - repository namespace 47, 83
 - XML namespace 83, 88
- Normalization 139

O

- Object check 37
- Object properties 215

P

- Package 45
- Partner Interface Process 154
- Payload 70, 163
- Payload-based routing 141
- PIP 154
- PMI 179
- Process integration content 32, 44, 49
- Processing log 37
- Product 44
- Proxies 65, 71, 88, 169
 - ABAP client proxy 73
 - ABAP server proxy 74
 - client proxy 71
 - regeneration 72
 - server proxy 71
- Proxy call 170
- Proxy generation 84
- Proxy runtime 20, 169, 170
 - error handling 172
- Proxy server 174

Q

- qRFC 167
- qRFC inbound scheduler 165
- Quality of service 164, 170

R

- Receiver agreement 127, 132

- Receiver determination 132, 233
 - For Integration processes 199
- Receiver-dependent routing 141
- Receiving IDocs 223
- Release transfer 48
- Representation 135
- Request 98
- Response 98
- RFC adapter 149
- RFC-XML 149
- RNIF 154
- RNIF adapter 148
- Role 53, 55
- RosettaNet 153
- RosettaNet Implementation
 - Framework 154
- Round-robin algorithm 166
- Runtime constants of the mapping
 - runtime 105
- Runtime Workbench 179

S

- SAP Alert Management 197
- SAP Exchange Infrastructure 15
- SAP interfaces 79
 - import 79
 - imported objects 80
- SAP NetWeaver 13
- SAP system ID 150
- SAP Web Application Server 14
- SAP Web AS → SAP Web Application Server
- Scenario → Configuration scenario
- Search help 38
- Secure sockets layer 147
- Send context 200
- Send step 199
- Serialization context 165
- Services 231, 250
- Shared collaboration knowledge 19
- SOAP-messages 177
- Software catalog 118, 213
- Software component 44
- Software component version 49, 214, 243
 - ABAP proxy generation 73
 - customer-specific 87
 - For mappings 99

- Importing 46
- Software unit 53
- Sort container contents 225
- Specific configuration data 163
- Stateful processing 185
- Stopping criterion 221
- StreamTransformation 104
- Switch 211, 225
- System error 172
- System landscape 213
- System Landscape Directory 44

T

- Technical SAP Systems 118
- Third-party systems 82, 118
- Top-Down approach 242
- Trace 168
- Transformation 212
- Transport targets 160
- tRFC 167

U

- UCCnet adapter 240, 252
- UCCnet Data Pool Service 239

V

- Value mapping
 - Mass filling 136
- Value mapping context 96
- Value mapping table 96, 136

W

- Warranty claims scenario 207
- Web service 68, 175
 - enhanced 177
 - Logischer Port 176
- Web service runtime 74, 175
- WebSphere message queue 241
- Where-used list 37
- Workflow 187
- WSDL 67
 - export 68
 - external definition 81

X

- XI message protocol 20
- XI runtime 175
- XML 20

- XML namespace 83
 - data type enhancement 88
- XML Schema 69, 70
- XPath 89, 105, 258
- XSD editor 69
 - columns 70
 - occurrence 71
- XSLT 105
 - Java enhancement 105
- XSLT mapping
 - XSD export 69