Applies to:
SAP NetWeaver Process Integration 7.1 including SAP enhancement package 1

Summary
This document provides both an introduction to the key concepts of SAP NetWeaver Process Integration and an overview of the tasks and tools that come into play in integration projects. It is targeted at beginners wanting to get involved in the topic and experts already involved in real-life integration projects and who need a handbook to help them stay ahead.

Author: Peter Gutsche
Company: SAP AG
Created on: 30 June 2009
# Table of Contents

1. **Introduction** ........................................................................................................... 4
   1.1 What You can Expect from this Document .............................................................. 4
2. **Basic Concepts** ......................................................................................................... 4
   2.1 Integration of Processes .......................................................................................... 4
   2.2 Mediation ................................................................................................................ 5
   2.3 Decoupling Business Semantics from Implementation Details ................................. 6
   2.4 Phases of an Integration Project ............................................................................. 6
   2.5 Overview of Tasks and Tools .................................................................................. 9
   2.6 Mediation Capabilities of SAP NetWeaver PI ......................................................... 10
3. **Installing and Configuring the Software** ................................................................. 14
4. **Designing Integration Content** ................................................................................ 16
   4.1 Introduction ............................................................................................................. 16
   4.2 Defining Software Component Versions ............................................................... 17
   4.3 Defining a Process Model ...................................................................................... 21
   4.4 Defining Interface Objects .................................................................................... 27
   4.5 Proxy Generation ................................................................................................... 31
   4.6 Defining Mappings ............................................................................................... 32
   4.7 Using Predefined Integration Content ................................................................... 33
5. **Configuring Integration Content** ............................................................................. 35
   5.1 Introduction ............................................................................................................. 35
   5.2 Overview of Tasks and Tools ................................................................................ 38
   5.3 Describing the System Landscape in the System Landscape Directory .................. 41
   5.4 Defining the Communication Components and Channels (Adapters) ..................... 42
   5.5 Configuring Integration Server-based Communication .......................................... 43
   5.6 Configuring Local Message Processing on the Advanced Adapter Engine ............. 48
   5.7 Model-based Configuration ..................................................................................... 49
   5.8 Integration Directory Programming Interface ...................................................... 52
6. **Operating SAP NetWeaver PI** ................................................................................ 52
   6.1 Overview of Administrative Tasks ......................................................................... 52
   6.2 Overview of Administrative Tools ......................................................................... 52
   6.3 Software Logistics ................................................................................................ 55
   6.4 System Management ............................................................................................. 56
   6.5 Troubleshooting ..................................................................................................... 56
7. **Mapping** .................................................................................................................. 56
   7.1 Mapping Programs ................................................................................................. 56
   7.2 Overview of Mapping Objects in the ES Repository .............................................. 58
   7.3 Operation Mapping ............................................................................................... 59
   7.4 Message Mapping (Graphical Mapping Editor) ..................................................... 60
   7.5 Advanced Mapping Techniques ........................................................................... 62
   7.6 Mapping Examples ............................................................................................... 66
8. **Integration Processes (ccBPM)** .............................................................................. 66
   8.1 Introduction ............................................................................................................. 66
   8.2 Designing Integration Processes ............................................................................ 67
   8.3 Configuring Integration Processes ......................................................................... 71
8.4 Best Practices and Examples ................................................................................................................... 71
9       Routing........................................................................................................................................................... 73
  9.1   Content-Based Routing.................................................................................................................................... 73
  9.2   Dynamic Routing........................................................................................................................................... 76
  9.3   Message Split ................................................................................................................................................ 76
10      B2B Integration ............................................................................................................................................. 79
  10.1  B2B Integration Design.................................................................................................................................. 79
  10.2  B2B Configuration......................................................................................................................................... 80
  10.3  B2B Integration Based on Industry Standards ........................................................................................... 83
11      Security ......................................................................................................................................................... 86
  11.1  Secure and Reliable Technical Landscape of SAP NetWeaver PI .............................................................. 86
  11.2  Secure and Reliable Messaging.................................................................................................................. 86
  11.3  Security Standards for Web Services ........................................................................................................ 87
  11.4  Restricting Access to Runtime Environment to Specific (Service) Users..................................................... 88
  11.5  Configuration Examples for Back-End Settings.......................................................................................... 89
12      Services Registry ............................................................................................................................................ 89
  12.1  Introduction................................................................................................................................................... 89
  12.2  Publishing Service Definitions into the Services Registry ........................................................................... 90
  12.3  Discovering Services in the Services Registry ........................................................................................ 92
  12.4  Additional Tasks........................................................................................................................................... 93
  12.5  SAP Enterprise Services Explorer tool for Microsoft .NET ......................................................................... 93
13      Web Services Reliable Messaging ................................................................................................................ 93
  13.1  Connecting the Integration Server to WSRM-Enabled Systems .............................................................. 94
  13.2  Setting Up Direct Communication between WSRM-Enabled Systems ................................................... 94
14      Appendix ....................................................................................................................................................... 96
  14.1  SAP NetWeaver PI Architecture and Tools................................................................................................. 96
  14.2  Service Provisioning...................................................................................................................................... 96
  14.3  Setting Up High-Volume Scenarios ........................................................................................................... 97
  14.4  Additional Connectivity Options ................................................................................................................ 98
  14.5  Glossary ....................................................................................................................................................... 99
  14.6  Further Reading and Training .................................................................................................................... 105
1 Introduction

1.1 What You can Expect from this Document

This document provides both an introduction to the key concepts of SAP NetWeaver Process Integration (SAP NetWeaver PI) and an overview of the tasks and tools that come into play in integration projects. It is targeted at both beginners wanting to get involved in the topic and see their first example scenario running, and experts already involved in real-life integration projects and who need a handbook to help them keep their orientation not lose the central theme.

The document is organized in such a way that, if printed out, it can be used as a self-contained reference for getting a basic understanding of SAP NetWeaver PI.

In addition to this, when used online the handbook serves as the first point of entry to those parts of the SAP Library documentation that cover the key concepts in detail, as well as the procedures for performing the necessary tasks end-to-end. Links to key documentation chapters in each section point you to those parts of the documentation where you can dig deeper into the topic touched on in the section.

In detail:

- Chapter 2 covers the basic concepts in short.
- Chapters 3 – 6 cover the basic concepts in detail and provide a description of the most important tasks that come into play in an integration project.
- Chapters 7 - 13 cover specific concepts in more detail than in the preceding chapters.

Note: The concepts and tasks described in this document correspond to release SAP NetWeaver PI 7.1 including SAP enhancement package 1. It is stated explicitly if a specific feature is not available before SAP enhancement package 1 for SAP NetWeaver PI 7.1.

Corresponding notes are added in blue boxes to give an impression of customer scenarios that use a specific concept of SAP NetWeaver PI.

2 Basic Concepts

This chapter introduces the key principles and basic concepts that form the three main phases of SAP NetWeaver PI. These phases constitute the main framework along which all concepts are explained in more detail throughout this document.

2.1 Integration of Processes

SAP NetWeaver PI is SAP’s implementation of Service-oriented Architecture (SOA) middleware and facilitates the integration of business processes that span different departments, organizations, or companies. We will start by introducing the term process component which will accompany us throughout this handbook. A process component is part of the value chain of a business application or a business process. If we assume that a business application ranges over different departments of one company, then a process component usually represents one part of the process that is performed in one department. The following figure displays an integration scenario and shows the separation of a business application into its process components (blue icons), as well as the connections between the process components. In the example outlined in the figure, the process components run in three different departments of a company: A, B, and C. Process components can run on different systems, can be hosted in different departments of a company, or can be implemented in completely different companies that have a business relationship to each other. The process components exchange data with each other and thereby ensure that the value chain of the business process as a whole is maintained.
The focus of SAP NetWeaver PI is not on the inner life of the individual process components or how the business logic is implemented within a process component but rather on how the process components exchange data with each other. Process integration is all about the choreography of data exchange between process components.

2.2 Mediation

Technically, the business logic of different process components in an integration scenario is implemented on different systems. Let us assume that the systems involved in an integration scenario communicate directly with each other. For example, if the process components run on different SAP systems, one SAP system calls another using a remote function call. We call this kind of communication “point-to-point” or direct communication. However an upgrade to one part of the system landscape would, for example, entail that all individual connections that are affected also have to be adapted as part of the upgrade. In the case of large system landscapes, this approach could easily get out of control since the number of connections grows to the square of the number of systems.

However, consider a situation where a central instance interconnects the systems as a communication hub or data hub. We call this type of communication mediated communication and refer to the data hub as the integration broker. With a central instance interconnecting the systems you then have the option to have all integration-relevant information accessible at one central location. In contrast to the point-to-point scenario where there is a “spaghetti-like” arrangement of connections, in a mediated scenario the number and arrangement of connections remains manageable.

The following figure illustrates the difference between mediated and point-to-point communication:
Mediated communication based on an integration broker is executed by exchanging XML messages. Accordingly, in the context of SAP NetWeaver PI we usually speak of message-based integration. The messages contain the business data exchanged between the systems involved in a cross-component process. The message protocol of SAP NetWeaver PI (which the integration broker can process) is based on the W3C standard SOAP Messages with Attachments (see also Messages).

Note: While we do cover direct or point-to-point communication (see Setting Up Direct Communication between WSRM-Enabled Systems), the main focus of this handbook is on mediated communication.

2.3 Decoupling Business Semantics from Implementation Details

The preceding sections have already set out the basic concept. If we assume the different parts of a cross-system business application and their interactions to be “hard-coded” on the individual systems the process spans, then every change at the technical implementation level (such as changing a server address) would entail a change of the whole business process. This is time-consuming, error prone, and does not scale for complex business processes and large system landscapes. Therefore, one basic principle is to decouple the business semantics from the technical details of the concrete system landscape. Business semantics are, for example, the business flow of a process and its separation into individual process components, as well as the structure of exchanged data. These aspects of a business process are merely determined by business considerations rather than by details of the implementation or of the concrete system landscape.

2.4 Phases of an Integration Project

Based on this decoupling, it is possible to describe the integration-relevant aspects of a business process at an abstract level first – irrespective of the details of a particular system landscape. We call the corresponding phase of an integration project the design time. At design time, those parts of a business process can be specified that are independent from any technical details which are implementation-relevant or system landscape-relevant. We have already introduced the integration scenario as a high-level description of the integration at design time and we will continue to use this term in the following discussion.

In a later phase – at configuration time – the integration scenario will be configured to run in a specific system landscape. You can consider one and the same integration scenario to be deployed on completely different system landscapes. For example, in one case there is a material management integration scenario that spans only few systems within a midsize company, whereas in another case the same integration scenario spans several hundreds of systems located in the different departments of a large enterprise. The same scenario in this case involves the execution of the same business logic - just on a completely different scale. The scenario is finally executed at runtime and can be monitored by an administrator.

The following figure illustrates the relationship of the design time and configuration time view:
Figure 3: Integration scenario (design time view; left) and assignment of process components to systems of the actual system landscape (configuration time view; right)

As an example, the figure shows the systems of the actual system landscape where the business logic of process components 1 and 2 is implemented: Process component 1 is deployed on systems 1a and 1b, whereas process component 2 is deployed on systems 2a, 2b, and 2c. Resulting from this, the communication between two process components is broken down to communication between the systems mentioned above at runtime, whereas the communication is mediated by an integration broker.

The three phases introduced here can be considered to be phases of an integration project: They form the basic framework for the detailed description of the concepts in this handbook.

### 2.4.1 Design Time

At design time, an integration developer designs the integration-relevant aspects of a business application at an abstract level, independent from any implementation-relevant details.

The following aspects of a business process can already be specified at design time:

- The process flow and its separation into individual process components (integration scenario and further process models to be discussed later-on)
- The interfaces that determine the data exchange between process components
- The detailed structure of the data – of the messages – that is being exchanged
- The mapping or transformation of data structures on both sides of a connection

The following figure gives an overview of these entities. On the left side it shows a complete integration scenario spanning a huge set of process components and connections; on the right side, the details of one single connection between two process components are shown:
Figure 4: Integration-relevant aspects of a business process that can be specified already at design time

We will explain in detail the design time-relevant concepts and procedures in the chapter Designing Integration Content.

2.4.2 Configuration Time

At configuration time, an integration expert (for example, an integration consultant) configures the integration scenario specified at design time for a specific system landscape to enable the scenario to run in this system landscape.

The first configuration task is to identify, the “players” of the game at runtime – the systems that actually communicate with each other – and relate them to the corresponding process components.

The following figure illustrates the relationship between the entities defined at design time (above, the communication of two process components including the entities introduced with figure 3) and those relevant at configuration time (below, for an exemplary system landscape). The different colors of the systems indicate the different technical characteristics the systems may be based on:
Based on this assignment, an integration expert specifies further details at configuration time on how the messages are to be exchanged between the systems:

- How the messages are routed by the integration broker from a sender system to one or multiple receiver systems
- How the individual systems (each may be based on different technical characteristics) can be connected to the integration broker (connectivity and adapters)
- Which security-relevant settings apply to the data exchange (for example, if messages are secured using digital signatures)

The configuration time-relevant concepts and procedures are explained in detail in the chapter Configuring Integration Content.

### 2.4.3 Runtime

The business process is executed in the system landscape at runtime, which means that the process is executed and messages are exchanged between the systems involved. An administrator can monitor the individual systems and the message flow.

The runtime-relevant concepts and procedures are explained in detail in the chapter Operating SAP NetWeaver PI.

### 2.5 Overview of Tasks and Tools

Referring to the sections above, integration projects can in principle be structured along the three phases Design, Configuration, and Runtime.
Note: Integration projects in “real life” might only cover a subset of the steps covered in this chapter. For example, a typical development project might only cover tasks that are related to the development of interfaces and mappings in the ES Repository (as related to the design time introduced before). On the other hand, you can also think of a project where an already designed integration scenario is set up for a customer-specific system landscape. In the latter project, you will mainly have to perform tasks related to the configuration time.

The following figure shows the phases and key tools that play the most important role in each phase:

![Integration Project Phases and Key Tools](image)

**Figure 6: Phases of an integration project and the corresponding key tools that come into play**

### 2.6 Mediation Capabilities of SAP NetWeaver PI

Before we dive deeper into the details of the three integration phases in the following chapter, let us now introduce the basic mediation capabilities of SAP NetWeaver PI.

The mediation capabilities describe how XML messages can be handled and processed by the integration broker:

- **Message Transformation (mapping)**
  Transforming the structure of the business data of a message during message exchange.

- **Message Routing**
  Forwarding a message sent by a sender system to one or more receiver systems.

- **Connectivity (Adapters)**
  Connecting the integration broker to sender and receiver systems based on completely different technical characteristics.

- **Integration Processes (ccBPM)**
  Cross-component Business Process Management (ccBPM) contains functions for enhanced service orchestration.
2.6.1 Mapping

In scenarios spanning different application systems, or even different organizations and enterprises, it is most likely that the structure of the data exchanged between two process components differs on both sides of a connection due to business-related reasons. To enable a seamless exchange of data, the data structures on both sides of a connection have to be transformed into each other.

Mapping determines the following aspects:

- How structure nodes (or elements) in a source structure are assigned to structure nodes in a target structure
- Which conversion rules apply for the transformation between source elements and target elements

Note: Mapping describes transformations at the level of the business data that is exchanged between process components. This can also include special formats for particular business entities, for example, the format of a time field in a message. Since data structures are merely based on business-relevant considerations, mappings between them can be defined at design time. Transformations at the level of the technical transport protocol are handled by adapters (as described under Connectivity).

The following figure illustrates a simple mapping step. Note that the figure illustrates what happens both at configuration time and at runtime, and shows systems connected to each other rather than process components. But keep in mind that mappings can already be defined at design time.

![Mapping Diagram](image)

**Figure 7: Simple mapping concatenating two fields of a source structure into one single target structure field**

The concepts and tools related to mapping are explained in detail under Mapping.

2.6.2 Routing

Routing covers all rules that define the flow of messages between different systems at runtime. SAP NetWeaver PI supports in particular routing that depends on the content of the exchanged message. For example, you can define a routing rule of the form that all messages with a specific value of one particular message field will be sent to a specific receiver system.

For example, the integration broker detects messages where the customer number field has a specific value and forwards them to specific receiver systems, which are intended to handle requests coming from the corresponding customer.

The following figure shows a scenario where a message is forwarded to three different receivers:
The concept of routing and more sophisticated routing options are explained in detail in the chapter Routing.

### 2.6.3 Connectivity

Connectivity is the capability to connect the integration broker to systems or applications that have different technical communication capabilities. Examples for technical communication capabilities are the HTTP protocol or a remote function call (RFC). The transformations of messages that are required at a technical level and that are necessary to connect the system to the integration broker are performed by adapters.

SAP provides a variety of adapters to connect the integration broker to sender and receiver applications that are based on completely different technical or application-specific protocols. The integration broker transforms each incoming message into an internal message format first before the message can be processed. This is done by an adapter at the inbound side (also referred to as: sender adapter). Depending on the characteristics of the receiver system, an adapter at the outbound side (a receiver adapter) then transforms the internal message format into the format or protocol the receiver can handle.

Note: Do not mix connectivity with mapping: connectivity implies transformations between the technical or industry-specific protocols of the connected applications. A technical "protocol" can be, for example, a simple file format, or an IDoc format. An industry-specific protocol can be RosettaNet or EDI. In contrast to that, mapping is the transformation of the business data in the payload of the message, which can include, for example, the transformation of one data field format (YYYYMMDD) into another (YYYY-MM-DD).

In particular, SAP NetWeaver PI provides connectivity to:

- Technical protocols such as JDBC, JMS, HTTP, and many more
- Industry-specific protocols, for example, RosettaNet or CIDX
- SAP applications that send or expect their data with IDoc and RFC

To ensure greatest possible spectrum of connectivity options, SAP provides a large set of own-developed adapters, and also accepts adapters developed by partners.

Additionally, you can develop your own adapters with SAP NetWeaver PI in case you do not find the adapter to fit your needs.
SAP NetWeaver PI is used extensively to integrate processes in heterogeneous system landscapes making use of the widespread connectivity capabilities. For example, Advanced Micro Devices, Inc. (AMD) is an American multinational semiconductor company that develops computer processors and related technologies for commercial and consumer markets that uses SAP NetWeaver PI 7.1 to integrate non-SAP systems: For this and more examples, see Customer Scenarios with SAP NetWeaver PI 7.1.

2.6.4 Cross-Component Business Process Management (ccBPM)

Cross-component Business Process Management (ccBPM) contains functions for enhanced service orchestration that are based on integration processes. An integration process is composed of a specific flow of steps – including the sending and receiving of messages – during which the status of the process is persisted on the Integration Server. In an integration process, you can define a specific level of process control. For example, you can specify how long an integration process must wait for further messages to arrive, or you can group incoming messages and then send them in a particular order. You can also define control structures, such as loops and process in branches that are independent of each other.
The concept is explained in detail in the chapter Integration Processes (ccBPM).

3 Installing and Configuring the Software

You have to install and configure SAP NetWeaver PI as a prerequisite for the following tasks.

The following sections summarize the different ways you can get to SAP NetWeaver PI 7.1 (including EHP 1), that depend on your initial situation (irrespective of whether you have already installed a previous version of the software or not).

3.1.1 Installing and Configuring SAP NetWeaver PI 7.1

We recommend you read the following blog on SDN for a brief overview of the installation procedure from scratch for SAP NetWeaver PI 7.1:

Quickly Install, Configure, and Run SAP NetWeaver PI 7.1.

The blog provides access to a quick start guide and a demo that take you through the complete installation and configuration procedure for a typical operating system and database step-by-step. Using the quick start guide, you can install and configure SAP NetWeaver PI 7.1 and run the first simple demo scenario in about 4 hours.

For the detailed installation procedure, check the installation guides for SAP NetWeaver PI 7.1.

You can find the installation guides on the following page on SAP Service Marketplace: Installation Information (SAP NetWeaver PI 7.1). Select the guide that covers your combination of operating system and database.

Installing SAP NetWeaver PI 7.1 Including EHP 1 (SMP login required)

This page lists all the guides and links required to perform a full installation of SAP NetWeaver PI 7.1 (including EHP 1), beginning with the planning phase and concluding with configuration.

3.1.2 Updating from SAP NetWeaver PI 7.1 to EHP 1 for NetWeaver PI 7.1

If you have already installed SAP NetWeaver PI 7.1, you can update to SAP enhancement package 1 (EHP 1) for SAP NetWeaver PI 7.1.

The following documents show how this works for a typical database and operating system:

- Update Guide to EHP 1 for SAP NetWeaver PI 7.1 (SMP login required)
This guide explains how to update your system from PI 7.1 to EHP 1 for PI 7.1. It also describes how to configure and run a simple process integration scenario.

- **Update Demo to EHP 1 for SAP NetWeaver PI 7.1** (SMP login required)
  This demo shows how to update your system from PI 7.1 to EHP 1 for PI 7.1. It also shows you how to configure and run a simple process integration scenario.

For the complete information on updating an SAP NetWeaver PI 7.1 installation to EHP 1 for SAP NetWeaver PI 7.1, see the [enhancement package installation guides](#).

### 3.1.3 Upgrading from SAP NetWeaver PI 7.0/XI 3.0 to SAP NetWeaver PI 7.1

If you have installed an earlier release of SAP NetWeaver PI, namely, SAP NetWeaver XI 3.0 or SAP NetWeaver PI 7.0, you can upgrade directly to SAP NetWeaver PI 7.1.

For the detailed upgrade procedure, check the upgrade guides for SAP NetWeaver PI 7.1. These guides are structured as checklists for the upgrade and guide you step-by-step through the procedure. You can find the upgrade guides on the following page on SAP Service Marketplace: [Upgrade Information (SAP NetWeaver PI 7.1)](https://service.sap.com/upgradeinformation). Select the guide that covers your combination of operating system and database.

The [Decision-making Factors when Moving to SAP NetWeaver PI 7.1 - Upgrade or New Installation with Phaseout how-to guide](https://service.sap.com/decision-making) discusses in detail the options available to you if you have an earlier version of SAP NetWeaver PI installed and want to move from the productive SAP NetWeaver PI 7.0/XI 3.0 landscape to SAP NetWeaver PI 7.1.

### 3.1.4 Calling the Process Integration Start Page

After you have installed SAP NetWeaver Process Integration, you can access the Process Integration start page (PI start page). From the PI start page, you have access to all tools of SAP NetWeaver PI. To get to this page, call transaction SXMB_1FR in the SAP system of the Integration Server.

**Note:** The URL of the PI start page in general is:

```
http://<host name of Integration Server>:<HTTP port>/dir
```

The screenshot shows how the SAP NetWeaver PI start page looks like.

![SAP NetWeaver PI start page](#)

*Figure 11: SAP NetWeaver PI start page*

More information on required user roles to access the tools: [Tool Access](#)
4 Designing Integration Content

This section covers the design time-relevant aspects of SAP NetWeaver PI in more detail. In addition to this, it also describes the main tasks that have to be performed in the design time phase of an integration project.

Note: The design time-relevant aspects are specified and stored in the Enterprise Services Repository (ES Repository). Those parts of the Enterprise Services Repository content (ESR content) that are used for designing the integration-specific aspects of business processes (and which form the basis of this chapter) are referred to as integration content.

Note that there is the option either to design the integration yourself from scratch or to use integration details already designed and delivered by SAP where you can modify this content according to your needs. We will go into the aspects of both approaches in this chapter.

Note: Both of these approaches normally come into play in real-life projects. A typical scenario would, for example, be that you use pre-defined content (and enhance it) to outline one part of the integration scenario, whereas another part has to be built completely from scratch. For the specific aspects that you have to consider when using predefined content shipped by SAP, read the section Using Predefined Integration Content.

Depending on the size of the company and the volume and complexity of the related business process, the development tasks might be distributed among different persons in a company – in the following for the sake of simplicity summarized as the integration developer.

4.1 Introduction

In this chapter, we assume that you use the top-down or model-driven design approach.

4.1.1 Top-Down Design Approach

Top-down design approach means: Firstly, you define at high level the overall process, in particular, its separation into individual process components and how these are connected with each other. This gives a bird’s eye view of the integration. You do this in a process model, as we will show below. Based on the process model, you then specify the other relevant integration content objects like interfaces, data types, and mappings in more detail.

These are the main tasks when you design integration content top-down:

1. Defining the products and software component versions
   Software component versions, based on software products, are the basic grouping entities of all objects created during design time. These entities are defined in the System Landscape Directory (SLD).

2. Designing integration content
   You perform the following tasks in the ES Repository:

<table>
<thead>
<tr>
<th>Design Task</th>
<th>Design Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling the integration-relevant aspects of a collaborative process – or how process components interact with each other</td>
<td>Process Models</td>
</tr>
<tr>
<td></td>
<td>You define process models to start with. To model the integration-relevant of a process, you use an integration scenario model, a process component interaction model, or a process integration scenario.</td>
</tr>
<tr>
<td></td>
<td>Based on the model, you create the corresponding integration content that specifies the integration in more detail (interface objects, mapping objects, and</td>
</tr>
</tbody>
</table>
communication channel templates (see below)).

<table>
<thead>
<tr>
<th>Specifying how a process component exchanges messages with another</th>
<th>Interface Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>There are different interface object types available that describe these aspects – from the communication mode of a message exchange down to the detailed data structure of a message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specifying how data structures are transformed into each other</th>
<th>Mapping Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>You use communication channel templates to specify those adapter settings for connectivity that are already known at design time.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pre-configuring connectivity</th>
<th>Communication channel templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>You store all the data of the software products used centrally in the SLD. For the design phase mostly relevant is the storage of software components as the central development packages for your design work in the ES Repository. Therefore, as a prerequisite for all development activities in the ES Repository, you have to define the necessary software components in the SLD.</td>
<td></td>
</tr>
</tbody>
</table>

3. Proxy generation in the corresponding back-end systems

The interface objects defined in the ES Repository are merely metadata that are independent from any programming language. Using proxy generation, you convert non-language-specific interface descriptions into executable interfaces.

Note: All subsequent tasks related to the programming of the business logic in the back-end systems, are out of scope of this handbook.

In the following sections, we cover each of these tasks and the related concepts in detail.

4.1.2 Design Time Tools

The following key tools are relevant at design time:

- Enterprise Services Repository (ES Repository)
  The ES Repository is the central tool for designing integration content. It provides an integrated modeling and design time environment that allows you to specify all relevant content objects seamlessly following a top-down or model-driven approach. From a tool perspective (the process model), you can navigate to the editors of these content objects.

- System Landscape Directory (SLD)
  You store all the data of the software products used centrally in the SLD. For the design phase mostly relevant is the storage of software components as the central development packages for your design work in the ES Repository. Therefore, as a prerequisite for all development activities in the ES Repository, you have to define the necessary software components in the SLD.

4.2 Defining Software Component Versions

As a prerequisite for all development tasks in the ES Repository, you need software component versions. You use a software component version to group together objects in the ES Repository that are shipped or installed together. Software component versions are originally created in the SLD (based on a software product) and then imported into the ES Repository.

For more information on how products and software component versions are related to each other, read Products and Software Components.
Note: If you have just installed and configured SAP NetWeaver PI but did nothing else, you will only see the SAP BASIS software component.

Procedure

To define a software component version in the SLD, proceed as described under Configuring, Working with and Administering System Landscape Directory.

The following figure shows the SLD start page:

![Figure 12: SLD start page](image)

To import the SLD-based software component version into the ES Repository, do the following:

Start the ES Repository from the PI start page. Enter user name and password and choose Process Integration as Application Profile. The application profile defines the set of ES Repository object types you can access during the session that have you logged into. Choosing application profile Process Integration ensures that you have access to all ES Repository object types that are relevant to perform design tasks in the context of integration projects.

Note: Application profile Service Definition is a filter to be used when working with SAP NetWeaver Composition Environment.

The following figure shows the user interface of the ES Repository with an ESR object (data type) opened in software component version SAP BASIS 7.10.
To import the software component version from SLD, proceed as described under Importing SLD-Based Software Component Versions.

Note: To develop objects for productive use, you have to use a software component version imported from the SLD (SLD-based). For test developments, you can also use local software component versions that do not have any counterpart in the SLD.

In the Edit Software Component Version dialog, you specify the attributes of the software component version according to your particular purposes. For more details, see Editing Software Component Versions.

The following figure shows the software component version editor:
These are typical settings for starting a development from scratch:

- **Objects are Original Objects/Objects are Modifiable**
  
  If you will only have one ES Repository in your landscape and want to create and change the original versions of your ESR objects in this software component version, leave both attributes checked.

  More information on how to use these attributes in more complex use cases: Software Component Versions

- **Use of Interface Objects**

  This setting influences the attributes available for the service interfaces developed in this software component version. Since service interface attributes have been changed with release SAP NetWeaver PI 7.1, it became necessary to separate developments based on “earlier” interfaces (developed with SAP NetWeaver PI 7.0 or SAP NetWeaver Exchange Infrastructure 3.0) and developments based on SAP NetWeaver PI 7.1.

  If you select SAP NetWeaver 7.0, the development of service interfaces is subject to the restriction that you can only define service interfaces that are compatible with SAP NetWeaver XI 3.0 or PI 7.0. This means that all new attributes invented with SAP NetWeaver PI 7.1 can no longer be used as service interfaces with more than one operation, or using interface patterns other than Stateless (XI 3.0-Compatible).

  Note: This setting has direct impact on the development of service interfaces for this software component version and cannot be changed later-on.

  More information: Usability of Interface Objects

- **Connection Data for Import from SAP System**

 专卖的条件设置用于从SAP系统导入的连接数据：
Should you want to import RFCs or IDocs from an SAP system, specify the details of the SAP system.

- **Uses External Documentation**
  Leave this attribute unchecked.
  This attribute enables ESR object documentation developed with SAP Solution Composer to be linked to the ESR object in the ES Repository. For the sake of simplicity, we do not cover this feature.

To subdivide the software component version further into semantic units (to structure your design objects), create namespaces as described under [Creating a Namespace](#).

### 4.3 Defining a Process Model

Top-down design of integration content begins with the development of a process model. In this section, we describe the key aspects of process modeling in integration projects.

#### 4.3.1 Overview of Model Types

The ES Repository supports process modeling with a variety of different model types. In this handbook, we only focus on the model types that are most important for outlining the integration-relevant aspects of collaborative processes:

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integration scenario model</td>
<td>This model type provides a high-level view on the integration. We have shown an integration scenario already in figure 1.</td>
</tr>
<tr>
<td>Process components interaction model</td>
<td>This model type allows you to specify the interaction between two process components in detail. Based on this model type, you can start specifying interface objects, mappings and channel templates.</td>
</tr>
</tbody>
</table>

Note: You can also use "classic" process integration scenarios to design the integration of components. This model type was already available in earlier releases of SAP NetWeaver PI, namely, SAP NetWeaver Exchange Infrastructure 3.0 and SAP NetWeaver PI 7.0, and is still supported in SAP NetWeaver PI 7.1. We will not cover this model type in this chapter; however, there is extensive documentation available about example scenarios based on this model type.

Note: You need to use this model type when designing scenarios using cross-component Business Process Management. Therefore, this model type is explained in the chapter Integration Processes (ccBPM). More information: [Defining Process Integration Scenarios](#)

#### 4.3.2 Defining an Integration Scenario Model

An integration scenario model allows you to specify the integration-relevant aspects of a collaborative process at the highest level. An integration scenario allows you to do the following:

- **Defining the process components and deployment units**
  A process component represents a part of the value chain of a business application. Assuming that the business process spans different departments of an enterprise, you can consider a process component to be part of the value chain that is performed in a department. First of all, you identify the process components that are relevant for your business process. You also identify the deployment units. A deployment unit groups all process components that will later be installed together on one system.
Defining the connections between process components

Now the integration comes into play: You specify which process components interact with each other. A connection can be specified further later-on with regard to whether it will be realized using mediation. But a connection on this level of detail can also represent a point-to-point interaction where no integration broker is involved.

An integration scenario model can contain all the interactions of a large business process that spans your whole enterprise, including communication with external business partners.

The following figure shows an integration scenario model and its elements:

![Integration scenario model](image)

*Figure 15: Integration scenario model*

More information: [Integration Scenario Model](#)

### 4.3.3 Specifying the Interaction between Process Components

An integration developer is usually not interested in the inner life of a process component; this is a matter for an application developer. An integration developer is more interested in the details of the connections between the process components - how the integration is achieved. You can use a *process components interaction model* to describe these aspects.

A process components interaction models shows the interaction between two process components in detail. You can consider a process components interaction model to be a detailed view of one single connection in an integration scenario model.
You can specify the following entities in particular in a process components interaction model:

- What are the interfaces for the message exchange and which operations are defined?
  How a process component transmits data to another process component is determined by a service interface. In a service interface, for example, the communication mode is defined (synchronous or asynchronous communication) and the direction: if the interface is responsible for a message to be sent out by the corresponding process component, it is an outbound interface. In contrast, if the interface determines how a message can be received by the process component, then it is an inbound interface. A service interface can contain one or more operations. Each operation represents a message that is either to be sent (in an outbound interface) or to be expected (in an inbound interface) by the process component.

- Which messages are exchanged between the process components (message type) and how are these messages structured (data types)?

- How are data structures transformed between different process components (mappings)?
  The interfaces for a data exchange between process components that belong to different organizations often differ from each other. In this case, specify a transformation of the data structures. The corresponding object is the operation mapping.

- Pre-configuration of adapters (connectivity)
  If at design time you already know some details about the connectivity of a process component with regard to the connection to another process component, you can specify these details in a communication channel template.

Note: A communication channel (as will be explained in detail in the section Defining the Communication Components and Channels (Adapters)) contains the configuration data for an adapter. The complete adapter configuration cannot be anticipated already at design time because it usually contains information such as the addresses and host names of systems. However, the type of adapter can in some cases already be anticipated at design time (for example, if a process component uses IDocs for communication). In these cases, you can create a communication channel template and assign it to the corresponding process component. Configuration of the integration will then be accelerated at configuration time.
The key feature of the modeling environment is that you can link a *modeled entity* (for example, for a service interface) with the corresponding editor where you can edit the service interface *object*.

Note: We also refer to this linking as an *object reference*. Object references are a general concept of the ES Repository: You can link from process models to interface objects, for example. Additionally, interface objects of different types can also refer to each other in a specific way.

The following figure shows which entities you can specify in a process components interaction model and which of these entities can be linked to corresponding ESR objects. For the highlighted elements, corresponding ESR objects can be defined (assignments), which means that the corresponding editors in the ES Repository can be opened (for example, the service interface editor and the mapping editor are shown below in the figure):

![Diagram of process components interaction model](image)

*Figure 17: Assignable elements of a process components interaction model*

For example, you can navigate from a modeled service interface in a process components interaction model to the corresponding interface editor. From the interface editor, you can then navigate to a message type editor and from there to a data type editor (top-down approach).

More information: [Process Components Interaction Model](#)

**Procedure**

We show how to start working with a process components interaction model using the modeling environment of the ES Repository.

For more details, read the SAP Library documentation under: [Modeling the Interaction Between Two Process Components](#).
These are the basic steps for creating a process components interaction model from scratch:

1. In the ES Repository navigation area, choose the software component version where you want to locate the process model, open the node Modeling, position the cursor on the Models folder, and choose New in the context menu.

2. To create a process components interaction model, select SAP ProComp interaction model as Model Type.

3. On the right-hand side of the graphical work area, you can select between different graphical entities to specify the process. Start by dragging an element for a process component into the work area.

4. You continue adding additional elements for the different entities that describe the process flow and the integration.

   The modeling entities available have already been introduced. For more modeling guidelines, see Modeling the Integration of Process Components.

5. Create the relevant assignments to ESR objects (for example, for the modeled service interfaces). You can assign an ESR object that already exists, however, if you start from scratch, we assume that the relevant ESR objects have to be created. You can create the relevant ESR object based on the model by positioning the cursor on the graphical element and choosing &lt;ESR object type&gt; Assignment ➔ Create Assignment in the context menu. This is shown in the following figure:
6. The editor for the ESR object opens: You can now continue creating the corresponding ESR object. Based on the process components interaction model, you have now created a set of ESR objects (for example, service interfaces, mappings, and so on) that are necessary to specify the integration. But these objects are still "empty": That is, only the object name, namespace, and software component version is specified. All other attributes still need to be specified.

4.3.4 Object References (Models)

To summarize, the following figure shows the most important object references that can be created from the process models to other ESR object types:
4.4 Defining Interface Objects

Interface objects provide all details on how a process component exchanges data with another one, for example, the mode of communication and the data structures.

4.4.1 Overview of Interface Objects

In this section, we provide an overview of the interface objects:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service interface</td>
<td>Defines a set of functions that is either provided by an application or used by an application; contains one or multiple operations. Each operation refers to one or more message types.</td>
</tr>
<tr>
<td>Message type</td>
<td>Defines the root element of a message and refers to a data type.</td>
</tr>
<tr>
<td>Data type</td>
<td>Defines a data structure.</td>
</tr>
<tr>
<td>External definition</td>
<td>Externally-defined data structure that is imported into the ES Repository.</td>
</tr>
<tr>
<td>Imported object</td>
<td>RFC or IDoc interface that is imported into the ES Repository.</td>
</tr>
<tr>
<td>Context object</td>
<td>Design object that can be used as an abbreviated expression for an XPath expression to address a specific payload element. Context objects are used in routing conditions (see Specifying the Routing and Mapping).</td>
</tr>
</tbody>
</table>

Interface objects can reference each other in a specific way. The possible object references are shown in the following figure and will be explained in detail below:
4.4.2 Service Interfaces, Operations and Message Types

As interface objects also encapsulate each other (as illustrated in the figure above), a top-down design approach is also feasible: Top-down design of interface objects starts with the definition of a service interface.

A service interface represents a set of functions that is either provided by an application (inbound service interface) or used by an application (outbound service interface). You can consider the set of functions a service interface represents as a subset of the functions implemented by a process component. However, a service interface in the ES Repository contains merely the metadata of a service, abstracted from any implementation details.

Service interfaces are based on Web Services Description Language (WSDL), an XML-based language for describing Web services and how to access them. However, when you create interface objects in the ES Repository, you do not need to be familiar with the syntax of WSDL because you can use the relevant editor to specify the interface attributes.

The category service interface attribute determines whether the service interface is an outbound or an inbound service interface.

Note: The category Abstract is only intended for communication with an integration process (cross-component BPM).

A service interface groups one or multiple operations. An operation represents the smallest, separately-callable function, described by a set of data types used as input or output. You can specify the Communication Mode for each operation. This attribute determines if the communication defined by the operation is synchronous or asynchronous. In a synchronous communication step, a response is expected for each call or request message that is sent out. In an asynchronous communication step, a request message is sent out but no response is expected.

You assign one or multiple message types for each operation – depending on the communication mode.

A message type defines the root element of a message. You use a message to exchange data between systems. A message type refers to exactly one data type that defines the structure of this data.

For a synchronous operation, you assign three message types: A request, a response, and a fault message type. A fault message type represents the message that is expected in case an error occurs. For an asynchronous operation, you only assign one request message type.

The following figure shows the role the different interface objects play for the message exchange as designed based on the process model (above); the example shows an asynchronous message exchange where a request message is sent from the service consumer to the service provider. As these terms will be used frequently in this document, the relationship of an outbound interface as a description of a consumer service and of an inbound interface as a description of a provider service is also shown:
The following figure shows a synchronous message exchange including a request and a response message being defined for the service interface operation:

Figure 23: Synchronous message exchange between a service consumer and a service provider

In a lot of scenarios, the application forces that a specific communication sequence and transactional behavior is maintained during message exchange. With SAP NetWeaver PI 7.1, the attribute interface pattern was introduced to make it possible to design this behavior in the ES Repository. An interface pattern describes the type of communication that is to be executed on the message when the interface is used. It determines what kind of operations can be defined for a service interface. The interface pattern that you select has an impact on the activities related to the programming of the business logic in the related back-end system (task of application developer).

For mediated communication using an integration broker, the interface patterns that fit most use cases are Stateless or Stateless (XI 3.0-compatible).

Note: The interface pattern Stateless (XI 3.0-compatible) is used by default for all interfaces migrated from earlier releases of SAP NetWeaver PI (namely, SAP NetWeaver PI 7.0 and SAP NetWeaver XI 3.0) to SAP NetWeaver.
PI 7.1 (called *message interfaces* in earlier releases). Additionally, this interface pattern is recommended for scenarios that use the common “technical adapters” such as the File/FTP, JDBC, JMS adapter. However, using this pattern limits the service interface to use only one operation.

The default pattern for developing new service interfaces is *Stateless*.

The interface pattern *Tentative Update & Confirm/Compensate (TU&C/C)* has been developed to improve the transactional behavior when using synchronous messages. The TU&C/C pattern ensures that - in cases of system or communication failure - one or more synchronous update calls in one transactional context are executed and ensure a consistent dataset on both sides of the communication.

The following figure shows a screenshot of the service interface editor.

![Service interface editor](image)

**Figure 24: Service interface editor**

More information:
- [How-to Guide - Best Practices for ES Repository Design Objects](#) (provides best practices on which interface pattern to select)
- [SAP NetWeaver Process Integration 7.1: ES Repository – Service Interfaces](#) (presentation; provides an overview of and introduction into interface pattern)

**Procedure**

To define interface objects, proceed as described in the following documentation chapters:
- [Defining the Service Structure with Interface Objects](#)
- [Service Interface](#)
- [Developing Data Types](#)
- [Service Enabling with SAP NetWeaver Process Integration 7.1](#) (article)
4.4.3 Data Type

A data type determines the data structure for a specific business entity, for example, an address. Data types are referenced by message types and therefore define the structure of a message.

As defined in the ES Repository, data types are based on XML Schema (also referred to as XSD).

Basically, data types can reference other data types which enables you to build up complex data types out of simpler ones. The basis for all data types are XSD types contained in XML schema and approved by W3C, (for example, xsd:string, xsd:decimal, xsd:integer). These data types define the properties of an element at a technical level.

To develop data types, there are the following basic alternatives:

- Developing data types according to UN/CEFACT Core Component Technical Specification (CCTS)
  
  This approach is most suitable for developing data types according to a governance process that ensures maximum re-usability of data types. Note, however, developing data types that comply with CCTS requires committees to govern the process.

  Core data types and aggregated data types are modeled according to CCTS. Aggregated data types must reference existing core data types or other aggregated data types, or both. A core data type is an XSD-based type that does not have any business-specific semantics yet.

  To provide a basis for modeling and design in the ES Repository, SAP has developed a huge set of global data types (GDTs). A GDT has business semantics and describes, for example, a basic business entity such as an address. GDTs are the basis for application-specific data types SAP-wide and can be used by customers to build their own data types.

  For more information, see Core Data Types and Aggregated Data Types

  Note: To browse the GDTs predefined by SAP, see the SAP Global Data Types Catalog (PDF 30 MB).

- Developing freely-modeled data types
  
  You can model a data type freely using the XSD Editor.

  For recommendations on which way to design data types, see the how-to guide SAP NetWeaver Process Integration Best Practices: Design.

4.4.4 External Definitions

If you would like to reuse existing XML definitions as message definitions described in either WSDL or XSD, you can import these XML definitions into the ES Repository, rather than re-enter them manually using the data type editor. You then encapsulate the imported definition in the form of an external definition.

4.4.5 Imported Objects

If you would like to reuse XML definitions of existing functions - RFCs or IDocs - so that you can call these functions using the Integration Server, you can import IDocs and RFCs into the ES Repository and make the interface signature known there.

4.5 Proxy Generation

A programming language-independent description of a service is available with the definition of interface objects in the ES Repository. To implement this service in the corresponding system connected to the integration broker, executable interfaces have to be created out of these abstract service definitions. These executable interfaces are called proxies. A proxy contains programming language-specific constructs, such as an interface with methods and parameters.

The business application in the system is then developed based on these proxies.
Note: The programming of the application is not within the scope of this handbook.

Depending on the direction of the service (outbound or inbound interface), you generate a consumer or a provider proxy. For ABAP and Java, proxy generation is performed by special transactions in the connected systems.

- ABAP proxy generation: AS ABAP, Enterprise Services Repository Browser (transaction SPROXY)
- Java proxy generation: SAP NetWeaver Developer Studio

### 4.6 Defining Mappings

In a mediated communication step of an integration scenario, the sending process component normally uses a data format and structure for sending out a message that is different to the one that the receiving process component can handle. Therefore, the data structure and format used by the sender has to be transformed into the structure and format that the receiver can handle. This type of transformation is called mapping.

You specify the corresponding transformation rules in the ES Repository – in the form of mapping objects. When you follow the top-down approach, you start with an operation mapping (for an outbound/inbound service interface operation pair).

The concepts and tasks are explained in detail in the chapter Mapping and under Mapping Messages to Each Other Using Mapping Objects.

### 4.7 Using Predefined Integration Content

You have the option to start an integration development project using integration content predefined by SAP. There is a lot of predefined content already available that can be reused and which helps customers to save time and effort in their integration development projects. Typically, customers do not use predefined content 1:1 without adapting it to their needs. A typical use case is that customers use data types, service interfaces, and mappings provided by SAP and build their own process model based on these entities, enriched by interfaces and mappings developed on their own. Another option is to use a predefined process model (and all underlying entities) where only one side of the communication is specified, and to specify the other part of the communication at the customers’ side.

The central location to browse for predefined integration content is the Enterprise Services Workplace (ES Workplace). You can access the ES Workplace in SAP Community Network at https://www.sdn.sap.com/irj/sdn/explore-es.

Note: The term enterprise service is used to emphasize the fact that the ES Repository contains services that were designed according to SAP’s SOA design principles. Technically, this term summarizes interface objects. In this document, we intend to name these objects in particular, that is, as service interface, message type, or data type. Integration content published on the ES Workplace was designed based on integration scenario models and process components interaction models. In addition to this content, SAP also provides integration content that was designed based on “classic” process integration scenarios. For an overview of all the content available, visit SAP Community Network at https://www.sdn.sap.com/irj/sdn/soa-modeling \(\rightarrow\) Integration Content.

Before customers can use and enhance predefined content, they have to download it from SAP Service Marketplace and import it into the ES Repository installed in their landscape. The corresponding location on SAP Service Marketplace is the SAP Software Distribution Center at http://service.sap.com/swdc \(\rightarrow\) Download \(\rightarrow\) Support Packages and Patches \(\rightarrow\) Entry by Application Group \(\rightarrow\) SAP Content \(\rightarrow\) ESR Content (XI Content).

Customer modifications of predefined integration content must not be executed in an imported SAP software component: They must be performed in a separate software component instead.

Note: This avoids conflicts with subsequent SAP software updates since changes to an SAP software component will be immediately overwritten when SAP software updates are imported.
Therefore, to be able to use predefined integration content provided by SAP, you have to create an own software component version for your developments. The new software component version has to include the SAP software component (that contains the predefined content) as the underlying software component version. To do this, define a “based-on” relationship between the new software component and the SAP software component.

Customer Experience

SAP NetWeaver PI 7.1 customers that have made extensive use of the predefined integration content provided by SAP are Energie Baden-Württemberg AG (EnBW) and Advanced Micro Devices, Inc. (AMD). For more information, see Customer Scenarios with SAP NetWeaver PI 7.1.

Procedure

To browse for and download integration content, perform the following steps:


2. Browse for the required integration scenario and – by clicking a connection – navigate to the required process components interaction model. From there, you can navigate to the underlying ESR objects (service interfaces, data types) you are interested in.

3. Make a note of the software component version for the ESR content objects you are interested in (as displayed in the Technical Data table).

4. Use the details of the software component versions to find the content package in SAP Software Distribution Center at [http://service.sap.com/swdc](http://service.sap.com/swdc). Select the Downloads tab. In the navigation area, choose Download → Support Packages and Patches → Support Packages and Patches - Entry by Application Group. On the Support Packages and Patches page, choose SAP Content, and on the following page choose ESR Content (XI Content), and select the content package (for example, XI CONTENT ROSETTANET).

![Figure 25: ESR content packages available in SAP Software Distribution Center](image-url)
Note: ESR objects (preferably interface objects) assigned to different process components within a model, are usually located in different software component versions. Mappings that describe the transformation between data types assigned to different process components are typically stored in a software component version that differs from the software components where the data types reside.

5. Download the content package and save it as .tpz files on your hard drive.

6. To import the content into the ES Repository into your landscape, proceed as described under Importing ESR Content.

To define a “based-on” relationship between software components in the System Landscape Directory, perform the following steps:

1. Open the new software component (which you have created for your own development) and choose the Dependencies tab.

2. Select the software component the new software component is based on and choose Define Prerequisite Software Component Versions.

3. As Context, select MetaDataRequest.

![SAP NetWeaver System Landscape Directory](image)

*Figure 26: Defining a “based-on” relationship between two software component versions in the SLD*

The detailed procedure is described under Defining and Removing Software Dependencies.
5 Configuring Integration Content

In this section, we cover the basic concepts for configuring integration content.

As already indicated in the section Configuration Time (figure 5), at configuration time you specify how messages have to be exchanged between the individual systems of the existing system landscape – in accordance with the process model specified at design time.

5.1 Introduction

Basically, at configuration time you have to perform the following tasks:

1. Determining the system landscape
   You identify the systems that are involved in the integration scenario.

   Note: At configuration time, the interaction of “abstract” process components is broken down into system-to-system interactions.

2. Determining how messages are exchanged between the systems
   You specify how messages should be exchanged between the systems in accordance with the integration specified at design time.

Task 2 is broken down into the configuration of individual system-to-system interactions. As you have to provide all the information necessary for the integration broker to handle the exchange of messages during configuration time, it is most natural to “take up” the position of the integration broker. This means, for each incoming message (which arrives at the integration broker), you have to determine what should happen with this message - for example, which receiver systems it is to be sent to, or how it is to be mapped.

The following figure illustrates this by highlighting the possible communication paths for a message sent from system 1a to the integration broker (in an interaction between process component 1 and process component 2). You can consider it as showing a subset of the communication illustrated in figure 3:

Figure 27: Message is sent from system 1a to the integration broker and forwarded from there to systems 2a-c
The following figure illustrates in more detail what happens with an incoming message:

![Process Flow Diagram](image)

*Figure 28: Configuration settings for an incoming message*

In particular, for an incoming message the following aspects have to be specified:

- **Inbound processing**
  Defines how the incoming message is to be transformed technically to the XML message format that the integration broker (or, in other words, the “PI runtime”) understands.
  
  Inbound processing might also include additional security-relevant aspects, for example, how to handle the signature of the incoming message, or how to decrypt the incoming message, if these special security standards have been applied by the sender of the message.

- **Routing**
  Defines which receivers the incoming message is to be forwarded to.
  
  The configuration of routing may also include routing conditions.

- **Mapping**
  Defines how the *business data* of the message is to be transformed with regard to a particular receiver (in contrast to the *technical* XML message format that is handled during inbound processing).
  
  For an incoming message sent to a particular receiver you select a predefined mapping from the ES Repository.

- **Outbound processing**
  Defines how a message should be transformed technically with regard to a specific receiver.
  
  Outbound processing again implies a technical transformation step: A transformation from the XML message format that the integration broker “speaks” to the protocol or standard that the receiver system can handle. Outbound processing may also cover additional security-relevant aspects, for example, how to sign or encrypt an outgoing message in case these security standards are agreed on with the receiver.

### 5.1.1 Integration Broker

At design time, we did not specify further the nature of the integration broker. At configuration time, you have to understand how the integration broker is characterized in more detail.

**Integration Server-based Message Processing**
By default (and also in earlier releases of SAP NetWeaver PI), mediated communication means that messages are exchanged using the Integration Server which is based on AS ABAP and AS Java (double-stack). This means the following: The Integration Server contains an Integration Engine that is based on AS ABAP and that provides the mediation services to process messages at runtime (mapping and routing).

Note: All SAP systems based on Application Server release 6.20 or higher contain a local Integration Engine that allows you to connect the system to the Integration Server. An application running on those systems can use a proxy to connect to the Integration Server. In this context, we also speak of connectivity based on the proxy runtime. All other systems – either SAP or third-party – connect to the integration broker using adapters. In this case, we speak of connectivity based on the adapter runtime.

Additionally, the Integration Server contains an AS Java-based Advanced Adapter Engine (AAE). If the AAE is also involved in the communication, it depends on the adapters used for inbound and outbound processing.

Note: Adapters run either on the Integration Engine or the AAE. The following adapters run the Integration Engine: IDoc, XI, HTTP, RNIF, CIDX, as well as the connectivity to WSRM-enabled systems (WS channel). All other adapters run on the AAE.

For example, if a JDBC (sender) adapter is used for inbound processing, the AAE is also involved in the communication to provide the required connectivity. The following figure illustrates this situation for the case where the AAE is connected upstream to the Integration Engine (providing the required connectivity at sender side):

![Diagram of Integration Server-based communication (including the AAE at sender side)](image_url)

**Figure 29: Integration Server-based communication (including the AAE at sender side)**

**Local Message Processing on Advanced Adapter Engine**

As of SAP NetWeaver PI 7.1, there is an additional option for processing messages locally on the AAE without involving the Integration Engine. This option ensures a considerably higher performance. However, using local message processing on the AAE provides only a sub-set of those mediation capabilities the Integration Server provides. For example, only a sub-set of adapter types can be used.

The following figure illustrates local message processing on the AAE:

![Diagram of local message processing on the AAE](image_url)
At configuration time, you have to decide on which option you choose.

5.1.2 Use of the Terms Outbound and Inbound

Since these terms occur quite often in this handbook, we would like to make some remarks on how the terms outbound and inbound are used in the context of SAP NetWeaver PI. When used in the context of design time objects, the terms outbound/inbound refer to the “perspective” of the application (or process component). When used in the context of configuration time objects, the terms outbound/inbound refer to the “perspective” of the integration broker.

For example: An outbound service interface (a design time entity) is a service interface whereby a message is sent out to another process component. In mediated scenarios, the message is sent to the integration broker and then sent from there to the other process component. Therefore, a message sent by an outbound interface is the incoming message as seen from the perspective of the integration broker. The incoming (or inbound) message (as used in the configuration time context) is then determined by an outbound interface implemented on a sender system. At configuration time, when speaking of inbound processing, we mean inbound with regard to the integration broker.

The following figure illustrates this relationship:

Figure 31: Usage of the terms outbound and inbound at design time and configuration time

5.2 Overview of Tasks and Tools

You structure, organize, and store configuration data in the Integration Directory as configuration objects. As a prerequisite for your configuration tasks in the Integration Directory, you have to define the system landscape in the System Landscape Directory (SLD).

In this section, we summarize the configuration tasks as well as the configuration objects that are used to structure the tasks and configuration settings. We will explain the concepts behind each configuration object in detail in the following sections.

This is the overview of the configuration tasks to be accomplished:

To specify the system landscape and communication capabilities of the systems involved, the following tasks are relevant:
a) Describing the system landscape in the SLD

b) Specifying which parties or systems communicate with each other and what their technical communication capabilities are (in the Integration Directory)

To do this, you define

- **Communication party**
  
  Represents a company unit that is to be involved and addressed in message exchange.

  Note: Communication parties will be explained in chapter B2B Integration.

- **Communication component**
  
  Represents an entity that can be used to address a sender or receiver of messages (typically a system).

- **Communication channel**
  
  Defines the rules of how to handle messages during inbound or outbound processing. In particular, you use a communication channel to define the type and configuration of the adapter used during inbound or outbound processing.

The following procedure for specifying the details of the message exchange in the given system landscape is dependent on the integration broker that is used.

- Specifying the message processing for **Integration Server-based communication**

  The following table lists the configuration objects used to specify the corresponding settings:

<table>
<thead>
<tr>
<th>Configuration Task</th>
<th>Configuration Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defining Inbound processing</td>
<td><strong>Sender agreement</strong></td>
</tr>
<tr>
<td></td>
<td>Defines the technical details for the inbound processing of a message for a particular sender/receiver pair, that is, how the Integration Server behaves towards the sender of a message. A sender agreement contains a reference to a (sender) communication channel.</td>
</tr>
<tr>
<td>Defining routing and mapping</td>
<td><strong>Receiver determination</strong></td>
</tr>
<tr>
<td></td>
<td>Specifies the receiver communication components (typically systems) and the corresponding routing conditions.</td>
</tr>
<tr>
<td></td>
<td><strong>Interface determination</strong></td>
</tr>
<tr>
<td></td>
<td>Specifies the corresponding inbound interfaces and may refer to a mapping (as specified in the ES Repository).</td>
</tr>
<tr>
<td>Defining outbound processing</td>
<td><strong>Receiver agreement</strong></td>
</tr>
<tr>
<td></td>
<td>Defines the technical details for outbound processing of a message for a specific sender-receiver pair, in other words, how the Integration Server reacts with regard to the receiver of a message.</td>
</tr>
<tr>
<td></td>
<td>A receiver agreement contains a reference to a</td>
</tr>
</tbody>
</table>
 Specifying **local message processing on the Advanced Adapter Engine**
You use one single object to specify all steps from inbound processing using routing and mapping right up to outbound processing: The **integrated configuration**.

As an additional option, you can also configure direct communication between systems based on the Web services runtime. We explain this concept in detail under Web Services Reliable Messaging. For the sake of completeness, we mention the relevant configuration object here, which is the **direct connection**.

To summarize:

- Communication parties, communication components, and communication channels represent the entities that communicate with each other and specify their communication capabilities.
- Receiver and interface determinations, and receiver and sender agreements (if the message is processed locally on the Advanced Adapter Engine: The integrated configuration) define the details of the message exchange between these entities.

For more information, see [Configuration Objects](#)

### 5.2.1 Object References (Configuration Objects)

The following figure shows the most important object references between configuration objects and between configuration and design objects:

![Configuration objects and how they relate to each other](image)

Figure 32: Configuration objects and how they relate to each other
5.3 Describing the System Landscape in the System Landscape Directory

As a prerequisite for all configuration activities, a description of the system landscape in the System Landscape Directory (SLD) has to be provided.

This is done in two phases:

- Defining technical systems for all systems that are actually installed in your system landscape
- Defining business systems for all systems that communicate with each other

Business systems are logical systems that play the role as senders or receivers of messages. Business systems can be SAP systems or third-party systems. Each business system has to be assigned to a technical system.

5.3.1 Separation of Business Systems and Technical Systems

Business systems represent "logical" systems, whereas technical system descriptions contain information about the technical details of a system. This is information such as the server address. When the message flow – that is, the routing – in a system landscape is defined (in the receiver determination), this is done based on business systems, that is, at the level of the "logical" systems. This separation of the routing definition from the technical details has the advantage that the configuration is independent from any changes to the technical details of the system landscape. If, for example, a server address is changed, this has no impact on the routing configuration.

Nevertheless, at runtime a message needs to be forwarded to a “real” installed system. That means that the server address has to be known at runtime. Therefore, in addition to the “logical” receiver determination (as mentioned above), a "technical" receiver determination is performed at runtime. This technical receiver determination is accomplished by a mapping of the business system to the underlying technical system. This dependency is already defined in the SLD since you always have to assign a technical system when creating a business system.

Note: As an example, this mapping of a business system and technical system shows up in the configuration data in the Integration Directory in the attributes of receiver communication channels that usually contain the server name of the receiving system. The server name is part of the technical system description that is maintained in the SLD. On the other hand, the communication channel is assigned to a (receiver) communication component which is mapped directly to a business system from the SLD.

The following figure shows a screenshot of the list of technical systems as maintained in the SLD (SLD start page → Technical Systems):

![Figure 33: List of technical systems in the SLD (example)](image-url)
When you define a business system (SLD start page \(\rightarrow\) Business Systems), consider the following aspects:

- For each business system, select the technical system that should be used as basis for the business system.
- Select the role of the business system (Application System for a system that acts as sender or receiver of messages in an SAP NetWeaver PI landscape) and assign the related Integration Server.

More information: Tasks in the System Landscape Directory

### 5.4 Defining the Communication Components and Channels (Adapters)

The following configuration tasks are all performed in the Integration Directory.

First of all, you define the entities that exchange messages with each other (the “players” of the game) - based on the system landscape described in the SLD. From a runtime point of view, you specify which systems act as senders and receivers of messages. For each business system involved in the message exchange (as defined in the SLD), you define a communication component in the Integration Directory.

Note: In this chapter, we describe how to set up the basic communication at configuration time and exclude, for now, more sophisticated use cases such as business-to-business (B2B) communication or cross-component Business Process Management. Therefore, it is sufficient for now to say that a communication component simply represents a business system. When explaining the more sophisticated approaches, communication components are used in a more generic manner, as will be described later. However, to understand the basic concepts of configuration, it is sufficient to restrict our explanation here to communication components of the type business system.

Additionally, you specify for a communication component how this component (or the system it represents) can communicate from a technical perspective, for example, if the system communicates by sending remote function calls or by sending an HTTP request. You use communication channels to define the technical communication capabilities of a communication component. The adapter type (which is one attribute to be specified when creating a channel) determines exactly the type of connectivity: You choose adapter type RFC for connectivity based on remote function calls, for example. The remaining attributes of a communication channel contain the configuration settings for the corresponding adapter.

The following figure shows a simple system landscape represented in the Integration Directory as four communication components interacting with each other using an integration broker:

![Communication Components and Channels](image-url)

*Figure 34: Communication components represent the entities that communicate with each other, communication channels their technical communication capabilities*
Note that the communication components identified can act as both senders and receivers of messages. Further configuration then consists of breaking down the picture to individual communication steps between senders and receivers of messages.

To define communication components and communication channels, proceed as described under Defining Communication Components and Defining Communication Channels.

5.5 Configuring Integration Server-based Communication

As already summarized in section Overview of Tasks and Tools, you need to specify the following configuration objects:

- Sender agreement
- Receiver determination
- Interface determination
- Receiver agreement

5.5.1 Configuration Object Keys

When you check the configuration objects in the navigation area of the Integration Directory, you will notice the following:

- Those objects that represent the “players” of the game and their communication capabilities (communication parties, communication components, and channels) can be identified by “usual” technical names.
- The objects that define the processing of a message (as listed above) are identified by complicated object keys.

An **object key** is a set of attributes required to identify an object. The figure below shows the elements the object key of a receiver determination is composed of:

*Figure 35: Object key of receiver determinations*

We explain an example object key for the receiver determination. To understand the object key structure in general, you need to know that an incoming message carries address information of the sender in its header.

For a receiver determination, the object key has the following structure:

```
<sender communication component> <outbound interface operation>
```
Note: In the examples shown in the figure above, the communication components are SAP systems with a specific client specified, and therefore have the form: <SAP system ID>_<client>, for example, QAE_100.

These entities identify the incoming message exactly: The name of the sender system identifies the origin of the incoming message, and the outbound interface operation defines details of the message, for example, the communication mode and the structure of the message.

That means the key of a receiver determination is constructed in a way that it definitely identifies a specific message coming from a specific sender system. At runtime, when the message arrives at the Integration Server, exactly that receiver determination will be evaluated whose key fits to the incoming message. For this message, the receiver determination attributes are applied and it is evaluated which receivers this message has to be sent to and which routing conditions apply.

For example, a receiver determination with the key ABC | PurchaseOrderRequest_Out.Create contains the configuration data for the Create message sent by the system ABC (using the interface operation PurchaseOrderRequest_Out).

The key of interface determinations, sender agreements, and receiver agreements is in principle structured according to the same logic. Therefore, when you create a configuration object in the Integration Directory, you first have to specify the key, and the key can no longer be changed.

Note: You also have the option to use wildcard characters to define a configuration object generically, that means, for all values of a specific key attribute. Generic definition of configuration objects enables you to minimize the work effort required when creating and changing configuration data. We will not go into the details of this here and but refer to the corresponding chapter in the documentation: Defining Configuration Objects Generically/Specifically.

5.5.2 Defining Inbound Processing

To specify the inbound processing, you define which (sender) communication channel (or sender adapter) is to be used to transform the incoming message into the message format of the Integration Server. You do this by defining a **sender agreement**. In a sender agreement, you also can define security settings (depending on the chosen adapter type).

The role of the sender agreement is illustrated in the following figure:

![Figure 36: Defining the inbound processing of a message coming in a sender agreement](image)

To create a sender agreement, you first have to specify the following object key attributes: The sender component (to identify where the message comes from) and the outbound interface (to identify which message the sender agreement applies to).
Note: Sender agreements are not mandatory in all cases. You only have to define sender agreements when using special sender adapters that are configured explicitly at the inbound channel of the Integration Server (for example, sender file/FTP adapters). You also have to define a sender agreement when you want to apply specific security settings for the processing of the incoming message.

For more information, see Defining Sender Agreements

### 5.5.3 Specifying the Routing and Mapping

The routing and mapping configuration is made up of the definition of **receiver determinations** and **interface determinations**.

- In receiver determinations you specify the receiver components (systems) of the message.
- In interface determinations you specify the receiver (or inbound) interfaces and assign the mapping that is to be applied.

The role of a receiver determination is illustrated in the following figure:

**Figure 37: Routing of an incoming message to three receiver systems 2a-c**

The role of an interface determination is illustrated in the following figure:

**Figure 38: Routing of an incoming message to a particular inbound interface of a receiver system**
You can also combine receiver and interface determinations with conditions. We explain this in detail under Routing.

Procedure

For each incoming message of a specific sender system (which is the key of the receiver determination), you define a receiver determination.

Note: You can either define the receiver determination for all operations of the outbound interface or for each operation separately (operation-dependently).

In the receiver determination (standard receiver determination to be explained first), you provide a list of receiver systems. For each receiver, you can then assign a routing condition. The following figure shows a receiver determination with two specified receiver systems SID_107 and SID_108; for the message to be sent to SID_108, a routing condition is defined:

![Figure 39: Receiver determination](image)

An interface determination is defined for a sender, an incoming message, and a specific receiver system. You can understand this structure of the interface determination key when you realize that at runtime first the receiver systems are determined (in the receiver determination step) and afterwards, the corresponding inbound interfaces are calculated for each of the determined receivers (in the interface determination step).

The following figure shows an interface determination for the `SendPurchaseOrderRequest` message sent from sender system SID_105 to receiver system SID_106:
You specify the inbound interface at receiver side in an interface determination by default and assign a mapping (chosen in field *Operation Mapping*) from the ES Repository.

More information:
- Defining Receiver Determinations
- Defining Interface Determinations

### 5.5.4 Defining Outbound Processing

To specify the outbound processing, you define which (receiver) communication channel (or receiver adapter) is to be used for an outgoing message (targeted at a specific inbound interface of a specific receiver system). You do this by defining a receiver agreement. In a receiver agreement, you also can define security settings (depending on the chosen adapter type). When creating a receiver agreement, you have to specify a receiver component, a receiver (inbound) interface, and a sender component.
In contrast to the definition of a sender agreement, the definition of a receiver agreement is mandatory in all cases. You can understand this by keeping in mind that for an outbound message (sent from the Integration Server to a receiver system) you have to provide information on how the Integration Server can communicate with the receiver system in any case. For many inbound communication cases it is implicitly known how the Integration Server can be reached by the incoming message.

More information: Defining Receiver Agreements

5.6 Configuring Local Message Processing on the Advanced Adapter Engine

Processing messages locally on the Advanced Adapter Engine is an alternative to using the mediation capabilities of the Integration Server. Local message processing on the Advanced Adapter Engine generally provides a higher message processing performance.

Note: Local message processing on the Advanced Adapter Engine is supported as of SAP NetWeaver PI 7.1.

Keep in mind that local message processing on the Advanced Adapter Engine implies limited mediation capabilities compared to message processing using the Integration Server.

The following adapter types cannot be used:

- IDoc
- XI
- HTTP
- RNIF
- CIDX
- WS

Additionally, the Advanced Adapter Engine only supports elementary communication scenarios for which the message receiver is already defined at the time of configuration and where message processing only requires a simple mapping (multiplicity 1:1). Last but not least, when using the Advanced Adapter Engine, you can only set up scenarios where the message is only to be transferred within one Adapter Engine from a sender to a receiver adapter.

Content-based routing on the Advanced Adapter Engine is only available as of SAP enhancement package 1 for SAP NetWeaver PI 7.1.
If you set up a scenario based on local message processing on the Advanced Adapter Engine, all considerations regarding inbound processing, routing, mapping, and outbound processing apply in exactly the same way as discussed in the preceding sections. However, in this case you can only define all these configuration settings in one single configuration object. In this object, the integrated configuration, you can then specify inbound processing, routing, mapping, and outbound processing on different tabs.

Note: As a prerequisite for configuring message exchange, you rely in the same way on communication components and channels as explained under Defining the Communication Components and Channels (Adapters).

The following figure shows a screenshot of the integrated configuration:

![Integrated Configuration Screenshot](image)

Figure 42: Integrated configuration object in the Integration Directory

The detailed procedure for configuring message exchange is described under: Executing Integrated Configuration.

### 5.7 Model-based Configuration

In this section, we show how you can bring together the information specified at design time with the activities to be performed at configuration time. Using a process model as configuration template is the best way to use synergies between design and configuration time activities. This allows a semi-automatic configuration which reduces configuration time, costs, and effort considerably. In this section we show how this works.

#### 5.7.1 Using Synergies between Design and Configuration Time

At design time (in a process model) you specify relations between process components. As you know from section Defining a Process Model, in a process components interaction model in particular you specify which interfaces, mappings, and possible communication channel templates determine the interaction between two process components in detail.

At configuration time, you basically map the interaction of process components to system-to-system-interactions. In the preceding sections, we explained how you do this manually: You manually defined the routing and the other processing details for each incoming message.
However, assume that at configuration time you use a process components interaction model as configuration template and first assign the involved process components to communication components (systems). Then you have all the information you need to derive all sender system/receiver system relations that are relevant for the interaction between the process components. In other words, you know the object keys of all the relevant receiver determinations, interface determinations, and sender and receiver agreements as explained in the section Configuring Integration Server-based Communication.

At this point, the model configurator comes into play: Based on assignments between process components and communication components (systems) for a specific model, the relevant receiver determinations, interface determinations, and sender and receiver agreements are calculated and automatically generated. This means that the tool calculates the object key, and checks if there already is a configuration object with this exact key. Depending on the result, it either creates the object or re-uses an existing one. All generated configuration objects are then grouped in a configuration scenario which is a grouping entity in the Integration Directory.

The following figure highlights how the design and configuration time entities are related to each other, how the model configurator comes into play, and how a receiver determination is generated out of the setting (as an example):

![Diagram](image)

*Figure 43: How the model configurator evaluates a receiver determination for the given example*

Using the model configurator saves you the trouble of manually creating all configuration objects. If you configure the integration for large system landscapes, it can quickly turn into a nightmare to manually find out all relevant configuration object keys.

After generating the objects, the configuration objects generated have to be further specified manually by adding those parts that cannot be automatically determined, for example, routing conditions or specific security settings in sender and receiver agreements.
Note: In the preceding sections, we have shown how to configure communication manually, that is, by creating each configuration object separately. We did this to explain the concepts behind each object type. However, we recommend you use the model configurator wherever it applies. Using the model configurator, you will be able to perform the configuration in a fraction of the time you would need to create all the objects manually.

You can use the model configurator with the following model types as input:

- Process components interaction model (or integration scenario as group of multiple process components interaction models)
- “Classic” process integration scenario

Note: The model configurator currently does not support scenarios including local message processing on the Advanced Adapter Engine. However, if you use the Advanced Adapter Engine, the configuration is required anyway due to the fact that you only need to define an integrated configuration object - this is not a serious restriction.

For more information, see Using the Process Model as a Configuration Template

Procedure

You start the model configurator from the Integration Directory menu (Tools → Apply Model from ES Repository).

You have to decide which model type you would like to use as template and then select the model from ES Repository using input help. If you choose SAP Integration Scenario Model as type, all process components interaction models used by the integration scenario are taken into account.

The following figure shows the model configurator opened for a process components interaction model:

![Process components interaction model opened in the model configurator](image)

Figure 44: Process components interaction model opened in the model configurator
The graphical interface of the model configurator shows the process components of a process components interaction model as vertical lanes.

| Note: For technical reasons, the support of process components interaction models is only available as of SAP enhancement package 1 (SP3) for SAP NetWeaver PI 7.1. |

5.8 Integration Directory Programming Interface

An alternative approach for creating configuration content is to use a programming interface.

Using the programming interface, you can make mass changes in the Integration Directory which you would not otherwise be able to do using the user interface (or at least only with a very time-consuming manual procedure).

For more information, see Integration Directory Programming Interface

6 Operating SAP NetWeaver PI

In this chapter, we provide an overview of the most important administrative tasks at runtime when you operate an SAP NetWeaver PI installation.

Operating SAP NetWeaver PI covers both the technical operation of an SAP NetWeaver PI installation and the administration of all its technical components, as well as the execution of integration scenarios that you have set up based on SAP NetWeaver PI (using the tasks as described in the preceding sections).

6.1 Overview of Administrative Tasks

These are the main administrative tasks:

- Monitoring
  Covers monitoring of system landscape and message exchange.
- Software Logistics tasks
- System Management
- Troubleshooting

You find detailed information about the tasks and tools in the Technical Operations Manual for SAP NetWeaver, in particular under: Administering PI (Process Integration). We only highlight some key aspects in this chapter.

6.2 Overview of Administrative Tools

There are different tools you can use depending on the task and the component you would like to administer. The following figure provides an overview of the most important administrative tools:
6.2.1 SAP NetWeaver Administrator and SOA Manager

To administer the back-end systems involved in an integration scenario as well as the AS ABAP or AS Java part of the Integration Server, you can use the SAP NetWeaver Administrator or the SOA Manager, respectively.

- **SAP NetWeaver Administrator** is the administration tool for Java-based back-ends involved in integration scenarios.
  
  You can access the SAP NetWeaver Administrator by entering the following data in a Web browser:
  
  \[ \text{http://<host>:<port>/nwa} \]
  
  Here the following represents:
  
  - `<host>` the host where the Application Server Java is installed.
  - `<port>` the HTTP port of the Internet Communication Manager. It consists of `5<Java_instance_number>00`. For example, if the instance number of the Java instance is 60, the HTTP port is 56000.

- **SOA Manager** is the administration tool for ABAP-based back-ends involved in integration scenarios.
  
  You can access the SOA Manager by using the transaction code SOAMANAGER in the back-end system.

Using these tools, you can perform the following administrative tasks:

- Monitoring services locally in the back-end system
- Publishing services from the back-end system into the Services Registry (more information)
- Services Registry

Note: You can also use to configure services or groups of services locally in the back-end system.
6.2.2 Message Monitoring Using the Runtime Workbench

When you execute an integration scenario, you can monitor the systems and components involved at runtime, as well as the message processing. We also refer to this as to process integration monitoring.

Process integration monitoring includes tasks like message monitoring, component monitoring, and performance monitoring. You can find detailed information under Process Integration Monitoring.

For process integration monitoring, you can use the Runtime Workbench (for more information, see Process Integration Monitoring).

Procedure

To monitor messages from the Runtime Workbench, perform the following steps:

1. From the PI start page. Choose Runtime Workbench and log in with user and password.
2. Choose Message Monitoring.
3. Choose the component that mediates the messages (for example Integration Engine <name of SAP system of Integration Server>) choose Display.
4. Under Filter, specify filter criteria for the messages to be displayed (for example, an interval for the processing time) and choose Start.
5. Select a message and choose Message Versions.

The different versions of the message during message processing are shown (as shown in the following figure):

![Figure 46: Different versions of a message shown in a table](image)

6. Select a message version and choose Details.

Displaying the details of different message versions, you can see, for example, how a message is transformed through mapping (as shown in the next figure):

![Figure 47: Message monitoring shows message payload before and after mapping](image)
6.3 Software Logistics

Software logistics covers both tasks related to the upgrade of your SAP NetWeaver PI installation, and tasks related to the change management and transport of the design objects and configuration objects created in the ES Repository.

You find detailed information under Software Logistics. In this section, we provide an overview of the topic of change management and transport.

6.3.1 Change Management and Transport of Design and Configuration Objects

Change Lists for Design and Configuration Objects

You can create different versions of design objects in the ES Repository and configuration objects in the Integration Directory. When you save an object for the first time, you create its first version. The object (still in edit mode) is stored in a user-specific change list (tab Change List in ES Repository or Integration Directory). The version ID is only closed when you activate the object. A new version is created when you start editing the object, that is already activated. The ES Repository and the Integration Directory provide different options for version conflict management. You find more information on object versioning under Change Lists (object versions are treated the same way in both the ES Repository and the Integration Directory).

When you have connected the ES Repository or the Integration Directory to a transport system, active objects can be transported.

Transporting Design and Configuration Objects

To transport design and configuration objects, you can use the following techniques:

- Transporting Objects Using the File System
  Using this option, you export the objects from the initial system as a file and you then import this file into the target system.

- Transporting Configuration Objects Using the Change Management Service (CMS)
  CMS is a component of SAP NetWeaver Development Infrastructure that you can use to transport software changes.
  You must first install and configure CMS before you can use it for object transports.

- Transporting Objects by Using Change and Transport System (CTS)
  CTS is a set of tools in the SAP System for managing and transporting ABAP Workbench and Customizing changes made in systems in the SAP System landscape and distributed between these systems. The enhanced CTS enables you to transport Java objects and SAP-related non-ABAP applications in your system landscape, alongside ABAP objects.
  You must first configure CTS and the source and target systems of the transport.

For more information, see Transporting ESR Content and Objects of Integration Directory

Note: Objects are not shipped from the Integration Directory because they reflect the configuration settings in a specific system landscape. However, you can have different Integration Directories installed in your landscape when you differentiate between test landscapes and productive landscapes, for example. In this case, you can transport from the test Integration Directory to the productive Integration Directory.

Versioning of Design Objects Using Software Component Versions

In addition to the object versioning options mentioned above, design objects in the ES Repository are assigned to different software component versions that reflect different versions of the content shipped by SAP.
Therefore, each design object is created in the context of a software component version. In contrast to that, configuration objects have no versioning corresponding to that at design phase, because configuration objects are not shipped.

For more information, see Organizing and Managing Content in ES Repository

You can transfer objects within the same ES Repository to another software component version.

For more information, see Transferring Design Objects

### 6.4 System Management

System Management includes the following task areas:

- **User Management**
  
  User Management of the SAP NetWeaver PI tools is based on the standard SAP NetWeaver [user management tools](#).
  
- **Configuration of basic technical settings using the Exchange Profile**
  
- **Logging and tracing for problem analysis**
  
- **Data archiving**
  
- **Backup/restore and recovery (based on the standard backup/restore and recovery tools of AS ABAP and AS Java)**
  
- **Periodical tasks like, for example, archiving or deletion of messages**

For more information, see System Management

### 6.5 Troubleshooting

In case of any problems during the operation of SAP NetWeaver PI, consult the very detailed Process Integration Troubleshooting Guide ([SAP NetWeaver PI 7.1](#)) (SMP login required).

In addition to that, check the SAP notes for the component BC-XI on a regular basis, in particular SAP Note [1060264](#) (SMP login required).

### 7 Mapping

Typically, a process component interacting with another process component in an integration scenario (in a mediated communication step) sends the business data in a different structure and format from that expected by the receiving process component. Therefore, the sent data structure has to be transformed or mapped to the structure the receiver can handle.

From the integration broker’s perspective, the incoming message (sent from a sender system) has to be transformed into an outbound message that is sent to a specific receiver.

Note: The term *mapping* describes transformations on the level of the *business data* that is exchanged between process components. This can include special formats for particular business entities, such as the format of a *time* field in a message. Transformations on the level of the technical transport protocol are handled by adapters (see Connectivity).

Since mappings are determined by business needs and describe data transformations on a *business* level, they can be defined at design time. Therefore, most tasks related to mapping are performed in the ES Repository. At configuration time, in the Integration Directory, you merely have to select the right mapping for a specific communication step (or interaction).

### 7.1 Mapping Programs

A mapping is implemented by a mapping program.
Basically, you use a mapping program to define:

- How structure nodes (or elements) in the source structure are assigned to structure nodes in the target structure
  Usually, elements with the same semantic meaning are assigned to each other. This part of the mapping is usually referred to as *structure mapping*.
- How the source element is transformed into a value target element

The following figure shows a simple mapping, where two fields in the source structure are merged into one field in the target structure (concatenation).

![Example mapping](image)

**Figure 48: Mapping merging two fields in the source structure into one field in the target structure**

More information: Mapping Messages to Each Other Using Mapping Objects

SAP NetWeaver Process Integration supports the use of different kinds of mapping programs:

- **Message mapping**
  With a *message mapping*, you map one message type to another by using a graphical editor in the ES Repository. A mapping program is generated from the graphical design.

- **Java program and XSLT (Java) program**
  These mapping programs are developed in Java and XSLT (eXtensible Stylesheet Language Transformations) respectively, and imported into the ES Repository as an archive.

  **Note:** XSLT (eXtensible Stylesheet Language Transformations) is a language that enables you to convert an XML document to another document. You can develop mappings with XSLT and import them to the ES Repository.

  More information: [XSLT Mapping](#)

  More information: [Java Mapping](#)

- **ABAP program and XSLT (ABAP) program**
  These mapping programs are developed using the ABAP Workbench. At runtime, these programs are executed on the ABAP Engine of the Application Server on which the Integration Server is running.

  **Note:** These kinds of mapping programs cannot be imported into the ES Repository and therefore are not shipped by SAP. They have to be developed in the SAP system at the customer’s site.

  More information: [Mapping Development with the ABAP Workbench](#)

There are several criteria that influence the decision about the most suitable mapping program type. These criteria include the programming skills of the mapping developers, as well as performance and software logistics. You can find recommendations and best practices for choosing the right mapping program type in the how-to guide SAP NetWeaver Process Integration Best Practices: Design.

For usability reasons, the design of message mappings is a good choice, because you can use the graphical editor.
## 7.2 Overview of Mapping Objects in the ES Repository

This section covers the design of mappings in the ES Repository.

**Note:** Since the development of Java and XSLT mapping programs is performed using external tools using standard programming languages, we will not cover this part of the topic. In this section, we show how you can either use existing Java and XSLT mappings, or create message mappings using the graphical mapping editor.

The following object types are available in the ES Repository for designing mappings:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operation mapping</strong></td>
<td>Assigns a mapping program for a pair of service interface operations. An operation mapping encapsulates the mapping program that has to be executed at runtime.</td>
</tr>
<tr>
<td><strong>Message mapping</strong></td>
<td>Mapping object created when you use the graphical mapping editor. A message mapping has to be referred to by an operation mapping.</td>
</tr>
<tr>
<td><strong>Imported archive</strong></td>
<td>Encapsulates an externally developed Java or XSLT (Java) mapping program. An imported archive has to be referred to by an operation mapping.</td>
</tr>
<tr>
<td><strong>Mapping template</strong></td>
<td>Contains parts of a message mapping that can be used as a copy template to create new message mappings. Mapping templates can refer to other mapping templates, thus enabling maximum reuse in mapping development.</td>
</tr>
<tr>
<td><strong>Value mapping group</strong></td>
<td>A value mapping transforms values of message elements. A value mapping group is a configuration object (Integration Directory) that contains mappings between different representations of one and the same object. Mapping programs (developed at design time) can refer to a value mapping table; however, since the actual values in many scenarios are not known at design time, the value mapping table content is maintained in the Integration Directory (see also Value Mappings).</td>
</tr>
</tbody>
</table>

The following figure shows the most important object references between mapping objects (also including the object references to interface objects):
7.3 Operation Mapping

You use an operation mapping to relate an outbound service interface operation with an inbound service interface operation. You can also relate IDoc and RFC interfaces with entities of the same type or with service interface operations.

This is illustrated in the following figure:

Figure 50: Assigning service interface operations with an operation mapping

For each operation mapping, you assign one or multiple mapping programs.

You define the assignment of the operations related to each other in an operation mapping, whereas in a mapping program you define the detailed transformation rules for the transformation of a source structure (representing the message sent by the outbound operation) into a target structure (representing the message received by the inbound operation).

The number of mapping programs or transformation rules you need to define for an operation mapping depends on the communication mode:

- **Synchronous communication**
  A synchronous operation (see section Defining Interface Objects) refers to a request, a response, and in some cases a fault message. Therefore, in general you have to define a mapping program for both request and response messages. If fault messages are used, you have to define an additional mapping program for the fault message.

- **Asynchronous communication**
  You only need one mapping program.

The following figure shows the operation mapping editor and the dropdown list box where you determine which type of mapping program to assign to the operation mapping:
An operation mapping encapsulates the used mapping program (either defined graphically by a message mapping or contained in an imported archive).

At configuration time, you have to select an operation mapping from the ES Repository for a specific communication step.

You do this in an interface determination (for integration using the Integration Server) or in the integrated configuration object (see Specifying Inbound Interfaces and Mappings; for local message processing using the Advanced Adapter Engine).

### 7.4 Message Mapping (Graphical Mapping Editor)

In a message mapping, you assign a source and a target message type (according to the source and target operation of the operation mapping the message mapping is assigned to). Using a graphical editor, you can define the mapping between the source and target message type.

Note: Java source code is generated from the message mapping and compiled in a .jar file. At runtime, the .jar file is executed.

The graphical mapping editor is composed of three main areas, as shown in the next figure:
Figure 52: Message mapping editor

The left area shows the structure of the source message assigned to the message mapping, the right area the structure of the target message. Below this is the data flow editor.

Note: If you have created the message mapping based on an operation mapping, the source and target structures are already loaded according to the operations you have chosen in the operation mapping.

You use the graphical editor to do the following:

- Design a structure mapping between the elements of the source and the target structure
- Apply a function to transform the source and target fields into each other, if necessary

A simple assignment between source and target fields is generally not sufficient.

In the data flow editor, you define target field mappings. A target field mapping describes how one or more source fields are mapped to a target field.

Note: A message mapping is constructed from individual target field mappings.

The following kinds of function can be used to specify target field mappings:

- Standard functions

  These functions are already available in the mapping editor. There are several kinds of standard function available in the mapping editor. For text mappings, for example, these include simple calculations or Boolean operations, and date format conversions.

  Note: In the example illustrated in the figure above, the two source fields Surname and FirstName are merged into the field PassengerName in the target structure using the standard (text conversion) Concat function.

- User-defined functions
You can define your own functions if standard functions do not fulfill your requirements.

More information: Data Flow Editor

### 7.4.1 User-Defined Functions and Function Libraries

You can create your own user-defined function in Java source code. A user-defined function created in a message mapping or mapping template is stored in a local function library belonging to the corresponding mapping object. To use a user-defined function in more than one message mapping or mapping template, you have the option to create the user-defined functions in function libraries in the ES Repository.

A function library is created and maintained as a separate design object in the ES Repository.

Note: A message mapping can only use function libraries that are defined in the same software component version as the message mapping, or in an underlying version.

### 7.5 Advanced Mapping Techniques

There are several advanced mapping techniques you can use to design and configure mappings.

Some of these techniques combine design time and configuration time features in a way that enables mappings to be used dynamically (value mappings or parameterized mapping programs, see below). You can find a detailed description of these techniques under Advanced Mapping Techniques. In this section, we provide an introduction to these techniques and summarize the most important aspects.

Best practices related to advanced mapping techniques are discussed in the how-to guide SAP NetWeaver Process Integration Best Practices: Design.

Note: As of SAP NetWeaver PI 7.1, there are numerous enhancements available that facilitate mapping design. For an overview of the new mapping techniques, read the SAP NetWeaver Process Integration 7.1: Mapping Enhancements Blog Series in SAP Community Network.

### 7.5.1 Multi-Mappings

In the preceding sections, we only considered mappings where a single source message is transformed into a single target message. This kind of mapping is also referred to as a 1:1 transformation.

A multi-mapping allows you to override this restriction. In particular, a multi-mapping gives you the following options:

- Designing a 1:n transformation for a message split
- Designing 1:n, n:1, and n:m transformations to be used in integration processes (cross-component Business Process Management)

Designing 1:n Transformations for Message Splits (Interface Determination)

You can use a 1:n multi-mapping to map a message to multiple different (and generally smaller) messages during logical routing (mapping-based message split). To set up a message split scenario, you have to perform the following steps:

- ES Repository: Create the necessary multi-mapping and assign it to an operation mapping
  More information: Developing Multi-Mappings for Message Splits
- Integration Directory: Select the operation mapping in the corresponding interface determination
  The inbound interface operations will be evaluated based on the multi-mapping.
  More information: Configuring Mapping-Based Message Splits

For more information on this technique, read section Mapping-Based Message Split.
Designing 1:n, n:1, and n:m Transformations to Be Used in Integration Processes

You can use a multi-mapping in a transformation step of an integration process. This is the summary of steps:

- ES Repository: Create the necessary multi-mapping and assign it to an operation mapping

  Note: It is a prerequisite that the message schemas for the messages to be mapped exist in the ES Repository and that they are assigned to asynchronous, abstract service interfaces. You can only use this interface type in integration processes.

More information: Developing Multi-Mappings for Integration Processes

- ES Repository: Create an integration process and enter the operation mapping in a transformation step in an integration process

More information: Transformation Step

For more information on integration processes, read section Integration Processes (ccBPM).

Procedure

To define a multi-mapping using a message mapping, perform the following steps:

1. Create a message mapping, select the source and target message (on the Definition tab).
2. On the Signature tab, set the occurrence for each message type (for example, 1... unbounded).
3. On the Definition tab, continue designing the message mapping using the graphical mapping editor explained above.

7.5.2 Value Mappings

If senders and receivers know the same objects under different names, value transformations are required.

Note: For example, a customer is identified in a sender system by a customer number, whereas in a receiver system the customer is identified by a name.

There are several options for performing value mappings:

- Using standard function FixValues
- Using standard function Value mapping

Using Standard Function FixValues

This is the easiest way to define a value mapping. In the target field mapping, you assign the standard function FixValues from the Conversions function group. Using this function, you can define value pairs. However, the value mapping defined by such a pair can only be used in the corresponding message mapping. Furthermore, this is a rather “static” option for defining a value mapping, since the value pairs have to be known at design time.

Using Value Mapping Tables from Configuration

A more flexible and “dynamic” way to define a value mapping is to use the standard function Value mapping (Conversions function group area). Using this standard function, you can refer to value pairs that are defined at a later point in time during configuration. To define the value pairs from configuration time, you use a value mapping group in the Integration Directory.

The advantages of this approach are that value mappings can be reused within different message mappings and values can be specified later at configuration time.
Note: In many cases, the names of objects are not known prior to configuration time.

To refer to the values (that are not already known at design time) in the message mapping, in the function Value mapping you use the key fields Agency and Schema, as shown in the screenshot below:

![Value Mapping Properties](image)

**Figure 53: Defining a value mapping at design time in the mapping editor**

Note: If you refer to a value mapping table created manually in the Integration Directory, you have to select [http://sap.com/xi/XI](http://sap.com/xi/XI) as Value Mapping Context. The value mapping context identifies the source of the value pairs.

At configuration time, you define the actual values of the objects by creating value mapping groups. A value mapping group is a configuration object in the Integration Directory and contains different representations of the same object. To enter the values that identify the objects in different frames of reference, you use the key fields Agency and Schema.

Note: Agency and scheme set a frame of reference within which an object can be uniquely identified. For more information, see Identifiers.

More information: Value Mapping (Integration Directory)

As an alternative to the manual creation and editing of value mapping groups, you can replicate value pairs from external data sources using a special interface. The interface objects are available as part of SAP predefined content in the software component SAP BASIS. For more information, see Value Mapping Replication.

### 7.5.3 Parameterized Mapping Programs

You can add parameters to a mapping program (enabling you to transfer values to the mapping program) in the following ways:

- At configuration time (in an interface determination in the Integration Directory)

  Note: This is another way to execute mappings dynamically, in the sense that the actual values are not known until configuration time (see also Value Mappings).

- At design time (using a transformation step in an integration process (ccBPM) in the ES Repository)
You can define parameterized mapping programs for message mappings, Java mappings, and XSLT mappings with Java enhancements.

To set up a scenario with a parameterized mapping program for a message mapping, you have to perform the following steps:

1. Define the parameters at design time. To do this, you have to perform the following steps:
   - In the message mapping (on the **Signature** tab), define the parameters to be used in the target field mapping.
   - In the operation mapping (that references the message mapping), create parameters (by choosing **Parameters**) and connect them with those of the message mapping (by choosing **Binding**).

   **Note:** You can set either **Simple Type** or **Adapter** as the **Category** for the parameter. The category **Adapter** is only relevant if you set up a mapping lookup scenario (see also Mapping Lookups).

2. Values can be entered for the parameters from the following locations:
   - **Integration Directory:** in the interface determination which uses the operation mapping
   - **ES Repository:** in a transformation step of an integration process

More information: [Parameterized Mapping Programs](#)

### 7.5.4 Mapping Lookups

Mapping lookups access values in a back-end system when the mapping program is executed (at runtime). By using a mapping lookup, a mapping program can call functions from other application systems while a mapping program is being executed on the Integration Server. The data transfer between mapping runtime and application systems is accomplished by means of a lookup programming interface (API). The API provides methods for accessing application systems using the RFC, JDBC, and SOAP adapter.

Therefore, to set up a mapping lookup scenario, you have to perform a combination of activities at design and configuration time:

1. At design time (in the ES Repository), you define the mapping program.

   You can define lookups for message mappings, Java mappings, and XSLT mappings with Java enhancements.

   At design time, you have to perform the following steps:
   a) Provide an import function parameter (of type **Adapter**)
   b) Implement the call to the application system (using the import parameter)

   More information: [Using the Lookup API in a Message Mapping](#)

   **Note:** When setting up lookups using the JDBC or the RFC adapter with a message mapping, you can perform this step graphically.

   More information: [Defining JDBC Lookups Graphically](#), [Defining RFC Lookups Graphically](#)

2. At configuration time (in the Integration Directory), you have to perform the following steps:
   a) Configure the corresponding adapter in a (receiver) communication channel
   b) Assign the corresponding operation mapping (that refers to the mapping program) in an interface determination

   You have to do this to ensure that the ID of the receiver channel is transferred to your mapping program at runtime.

More information: [Adding Lookups to Mapping Programs](#)
7.6 Mapping Examples

You can find examples of mappings in the ES Repository, software component SAP BASIS (the software component which is already available once you have installed SAP NetWeaver PI).

In software component version SAP BASIS 710, namespace http://sap.com/xi/XI/Demo/Agency, check the following message mappings:

- BookingOrder_Agency2Airline
  The fields for the first name (FirstName) and last name (SurName) of the passenger in the source data structure are merged into a single string (PassengerName) in the target data structure.

- BookingOrder_Agency2AirlineIDoc
  Maps a service interface operation to an IDoc structure.

- SplitMultipleBookingOrder_BPM
  This multi-mapping generates two messages out of one source message.

- BookingOrderDataInfo_Agency2AgencyReporting
  This mapping reduces the number of hierarchy levels in the target structure.

Customer Experience

Customers that make extensive use of the different mapping techniques provided with SAP NetWeaver PI 7.1 are, for example, Energie Baden-Württemberg AG (EnBW) and Swisscom IT Services AG. For more information, see: Customer Scenarios with SAP NetWeaver PI 7.1.

8 Integration Processes (ccBPM)

In the preceding chapters, we considered message exchange only for cases where messages are received and forwarded by an integration broker without any correlations between messages. Each interaction step is decoupled from the others: the integration broker processes an incoming message, forwards it to the configured receivers, and applies additional actions to the messages according to the configuration data. However, once the message has been sent, no status is held; the integration broker “forgets” the message. With cross-component Business Process Management (ccBPM), the message choreography capabilities of SAP NetWeaver Process Integration are extended so that a status is kept. In this section we explain how this works.

8.1 Introduction

You can use ccBPM to define an integration process. An integration process is composed of a specific flow of steps (including the sending and receiving of messages), during which the status of the process is persisted on the Integration Server. In an integration process, you can define a specific level of process control. For example, you can specify how long an integration process must wait for further messages to arrive, or you can group incoming messages and then send them in a particular order. You can also define control structures, such as loops and processing branches that are independent of each other. You can define conditions that control processing depending on the result of the condition. You can correlate messages with each other in order to ensure that messages that belong together are processed by the same integration process instance.

8.1.1 Overview of Tasks

To set up a scenario including ccBPM, you have to perform the following tasks:

1. Design an integration process in the ES Repository
An integration process is designed as a separate design object in the ES Repository using a graphical editor. To make sure that the integration process can send and receive messages, you have to embed the integration process in an overall scenario or model. The model type has to be a process integration scenario.

**Caution:** In contrast to the top-down procedure described under Designing Integration Content, when using integration processes you cannot use an integration scenario model or a process components interaction model as the overall process model. Instead, you have to use a “classical” process integration scenario.

2. Configure the integration process in the Integration Directory

At configuration time, you treat the application component containing the integration process as a single communication component. Based on this assignment, you configure the process integration scenario using the model configurator.

3. Execute and monitor the integration process

Integration processes are executed using a separate runtime component, the Business Process Engine, which comes into play for the execution of integration processes on the Integration Server.

More information: Defining Process Integration Scenarios
More information: Defining and Managing Integration Processes

### 8.2 Designing Integration Processes

To design an integration process, you can use a graphical process editor, which is explained below. Once you have designed the integration process, you have to create a process integration scenario and assign the integration process to it.

### 8.2.1 Specifying the Integration Process

The procedure for specifying the “inner workings” of an integration process (the integration process editor) is documented in detail under Defining and Managing Integration Processes. In this section, we summarize the main aspects.

#### Specifying Service Interfaces

To enable the integration process to exchange messages with other application components, you need service interfaces. For service interfaces used in integration processes, you have to select Abstract as the Category.

An abstract service interface is an interface that has no defined direction initially. Whereas outbound or inbound interfaces have an implemented interface in an application system as a counterpart, abstract service interfaces are only used by integration processes to send or receive messages. You can use the same abstract service interface to receive or to send. You cannot generate a proxy for this interface type.

**Note:** An integration process can only reference service interfaces from its own software component version.

More information: Process Signature

#### Specifying the Integration Process (Graphical Process Editor)

These are the main elements of an integration process:

- Container elements

  You define the data the integration process has to process (its “variables”) as container elements. Container elements can have the following types:

  - Abstract interface
For messages (to be used in receive or send steps) described by corresponding abstract service interfaces.

- Simple XSD Data Type
  For process control elements, such as counters.

- Receiver
  Used in a receiver determination step (see below).
  The actual list of receivers is determined at runtime based on a receiver determination in the Integration Directory.

You need appropriate container elements to specify the corresponding process steps.

- Configurable parameters
  You can define parameters whose values you can specify later, at configuration time. This gives you greater flexibility in handling the integration process.

- Process steps
  You construct the integration process out of different process steps. You can use different step types. For example, you use a send step to send a message to another integration process or to a business system.

When specifying a process step, you choose the corresponding container elements. For example, when you define a receiver determination step, you need a container element of the type Receiver to assign to the step.

For detailed information, see Step Types. However, note the following:

- Transformation steps are used to transform messages. Therefore, these steps refer to an operation mapping. You can use 1:n multi-mappings to split a message into multiple messages, n:1 multi-mappings to bundle multiple messages into a single message, or 1:1 mappings for message transformations (see also Multi-Mappings).

- In receive steps and send steps, you refer to the container elements for the corresponding abstract service interfaces.

- In a receiver determination step, you refer to the corresponding Receiver container element.

- In switch steps, fork steps, and loop steps, you can define conditions.

- User decision steps enable users to make decisions that influence the processing of the integration process. The intended user receives a dialog work item in the workflow inbox at runtime.

- Message correlations
  You use a correlation to join messages that belong together to the same process instance. In a correlation you define the message elements (for example, a customer number) by which messages are to be correlated.

- Exception Handling
- Deadline Monitoring
- Defining Sync/Async Communication
  You can define a "sync/async bridge" to enable the communication between a synchronously calling business system and an asynchronously called business system.

You define these elements using the graphical process editor. The following figure shows a screenshot of the process editor:
8.2.2 Embedding the Integration Process into a Process Integration Scenario

To enable the integration process to exchange messages with other (application) components, you have to embed the integration process into a process integration scenario as the overall model. In a process integration scenario, you use application components to model the communication partners. The concept is similar to that of a process components interaction model. However, there are significant differences between the two approaches:

- In a process integration scenario, you start with application components as basic entities that exchange messages with each other.
  
  An application component represents an overall application area, such as SAP SCM. Those “process steps” that exchange messages with each other are modeled as *actions* within an application component.

- In a process integration scenario, you do not model the service interfaces and mappings as separate graphical elements. Nevertheless, you can navigate from the process integration scenario editor directly to the corresponding editors, by clicking on a connection arrow in the process integration scenario for example. Insofar, process integration scenarios also support the top-down design approach.

You find the detailed description of the concept as well as modeling guidelines under [Defining Process Integration Scenarios](#).

These are the basic aspects you have to consider for ccBPM:
For the integration process (to be specified later in a separate editor), you insert a separate application component. In the process integration scenario, you have to model the parts of the integration process that are involved in message exchange interactions. You use actions to do this.

Note: Keep in mind that the application component defined for the integration process in the process integration scenario can be seen as the *signature* of the integration process. It displays those parts of the integration process that interact with other application components using message exchange.

You can navigate from the corresponding application component to the integration process editor to display or further specify the “inner workings” of the integration process.

The following figure shows how an integration process is embedded into a process integration scenario:

Figure 55: Process integration scenario using an integration process

### 8.2.3 Object References (Integration Process)

The following figure shows the most important object references related to integration processes:
8.3 Configuring Integration Processes

At configuration time, the integration process designed in the ES Repository acts as sender and receiver of messages and, therefore, is treated as a separate communication component. From a configuration time point of view, the integration process is nothing more than a “black box”.

To enable an integration process to be addressed as a sender or receiver of messages, you need to define a communication component of type integration process (or integration process component). You basically create a communication component, specify a name, and assign the integration process from the ES Repository. If you have defined Configurable Parameters, you can specify them further here.

Note: You do not assign a communication channel to an integration process component.

The subsequent configuration of the message exchange including the integration process component is then performed using basically the same procedure as already described under Configuring Integration Content and Configuring Communication Using the Integration Server.

Note: You cannot use integration processes when you configure local message processing on the Advanced Adapter Engine.

Since you need to assign the integration process to a process integration scenario, we recommend that you use the process integration scenario as a configuration template (model configurator).

8.4 Best Practices and Examples

There is a wealth of information material available showing step-by-step how to design integration processes. Here is a list of some key information sources:

- Tutorial: Defining an Integration Process
- Checklist: Making Correct Use of Integration Processes
- SAP NetWeaver PI Demo Exercise - With ccBPM (SAP EHP 1 for SAP NetWeaver PI 7.1)
  This document explains how to develop and test a simple process integration scenario including an integration process.
- Examples and Usage Cases
Describes simple example integration processes that are also part of SAP standard shipment (delivered in software component SAP BASIS).

**Customer Experience**

Energie Baden-Württemberg AG (EnBW) and Arla Foods are examples of customers that extensively use integration processes with SAP NetWeaver PI 7.1.

As additional examples:

A supplier for the automotive industry uses an integration process to collect messages sent from different systems (using a receive step). The messages are collected until a particular number is reached; the messages are then merged into a new message, which is sent to a receiver.

An online retail customer uses an integration process to receive web orders and to transform them into IDocs. In this customer scenario, a multi-mapping is also used to split the orders into individual IDocs.

For these and more examples, see [Customer Scenarios with SAP NetWeaver PI 7.1](#).
9 Routing

This section provides more insight into the concept of routing and introduces advanced routing concepts. The basics of routing have been described under Configuring Integration Content.

9.1 Content-Based Routing

In many business cases, it is necessary to define conditions with which the receivers of a message are determined during routing. For example, consider a condition in the following form: “If the value of a specific field in the message is x, then forward the message to receiver y.”

At configuration time, you can define conditions that depend on the content of the message. You can do this for both receiver determinations and interface determinations.

9.1.1 Content-Based Receiver Determinations

The following figure illustrates a typical use case for content-based routing.

![Diagram of content-based routing](image)

Figure 57: Basic example for content-based routing (from the PI Demo Examples)

The example illustrated in the figure above is described in detail for the SAP NetWeaver PI Demo Examples. In the example, a flight availability check request is sent from a travel agency to an airline. To ensure that the request is forwarded to the correct airline system, the routing condition is formulated as airline-dependent. The airline ID is contained in the payload of the message. The routing condition is as follows:

Send the message requesting the flight availability check to the airline Lufthansa if the field AirlineID in the message payload has the value LH.

To configure this behavior, you add a routing condition to a receiver determination. Generally, a routing condition has the following syntax:

\[
\text{<element in the message><Operand><value>}
\]

The element in the message is identified by an XML Path Language (XPath) expression. XPath is a language that allows you to address parts of XML documents. However, using the condition editor (see below), you do not need to worry about XPath syntax, since you can conveniently identify a message element by clicking through the message structure displayed in the editor.

Additionally, you can combine multiple conditions with logical AND and OR operators.

Procedure

To define a routing condition, in the receiver determination editor, call the input help for the Condition field for the corresponding receiver. The condition editor opens as shown in the following figure:
Figure 58: Condition editor for a configured receiver in a receiver determination

With the left operand, you specify the payload element of the incoming message upon which the routing to the specified receiver is to depend. In the right operand, you enter a value for the payload element. You choose a specific operator to link both operands.

To specify the left operand (the payload element), you can use an expression editor as shown in the following figure:
You can specify the payload element by choosing either an XPath expression or a context object.

- With the **XPath** option, you can select the payload element intuitively from the structure of the incoming message (which is defined by the outbound interface in the key of the receiver determination).

- Using a **context object**, you can select from the context objects that have been defined for the outbound interface. A context object is a design object that can be used as an abbreviated expression for an XPath expression to address a specific payload element.

Note: A context object has to be defined with the corresponding outbound interface in the ES Repository beforehand. So, if you already know at design time the payload elements upon which the routing is likely to depend, you can define the corresponding context objects in the ES Repository at the corresponding service interface.

A **routing condition for a specific receiver or a set of receivers** is called a **routing rule**. When you define a routing rule within a specific receiver determination, this condition is only valid for the given key (that means, for the specific incoming message of the given sender system). This kind of routing rule cannot be reused for other incoming messages or other sender systems. Therefore, it is called a **local rule** (as also shown in the screenshot above). You can define a local routing rule by first defining a reusable **receiver rule** independently of any combination of incoming message and sender system and, in a second step, assigning this receiver rule to a specific receiver determination.

You define a receiver rule like a local routing rule as explained above, as shown in the figure:
9.1.2 Defining Content-Based Interface Determinations

You can also assign a condition for each configured inbound interface (and mapping) in an interface determination. You do this using the same condition editor as for receiver determinations.

9.2 Dynamic Routing

When using routing conditions or routing rules in receiver determinations, the receivers of a message are determined at runtime by evaluating the condition (which depends on the content of the message). However, the names of the receivers have already been defined at configuration time in the receiver determination editor. Therefore, this kind of routing is still static with regard to the individual receiver names.

A dynamic routing option, which even includes the receiver names, is the enhanced (dynamic) receiver determination. In an enhanced receiver determination, you can configure runtime behavior to ensure that the names of the receivers are determined at runtime by a mapping program.

A typical usage case is if you do not yet know the names of the receivers at configuration time. However, note that if you want to implement a dynamic receiver determination, you also have to maintain a corresponding mapping (to be defined in the ES Repository). The mapping can be specified in such a way that it reads a list of receivers from a table or from the payload of the message at runtime.

9.3 Message Split

Using the routing capabilities of the integration broker (Integration Server or Advanced Adapter Engine), you can set up scenarios where a message is split into several fragmented messages at runtime, sent to the same or to different receiver systems.

Note: When using the Advanced Adapter Engine, these sophisticated routing options are not supported in releases prior to SAP enhancement package 1 for SAP NetWeaver PI 7.1.

To clarify the options, we briefly explain the basic scenarios that you can configure with an interface determination:

- Interface split
Mapping-based message split

**Interface Split**

By default, in an interface determination you specify one or more inbound interfaces for a given receiver system. For each inbound interface, you might also like to assign a mapping since the inbound interfaces are most likely different from each other.

The runtime behavior is illustrated in the figure below:

![Figure 61: Interface split](image)

**9.3.1 Mapping-Based Message Split**

When you have a larger message (for example, one message containing a large number of line items) to be split into several smaller messages with each message containing only a subset of line items of the large message, then you can do the following:

1. In the ES Repository, define a 1:n **multi-mapping** to specify the transformation of the large message into n smaller messages.

2. In the Integration Directory, define an interface determination with the following properties:
   - The interface determination key contains the source interface of the multi-mapping as outbound interface.
   - In the interface determination, select the multi-mapping from the ES Repository (in the **Operation Mapping** field).

   The target interfaces defined for the multi-mapping in the ES Repository are then calculated and displayed in the interface determination.

The figure below shows the behavior at runtime:

![Figure 62: Message split based on a 1:n multi-mapping](image)

Using this option, you can only configure a message split where the split messages are sent to different inbound interfaces of the **same** receiver system.

The reason for this is as follows: During runtime, the receiver determination step is performed prior to the mapping step. At the time when the multi-mapping is performed and the corresponding inbound interfaces are calculated (and the corresponding split messages are generated), there is no chance to do another “receiver split”. The resulting split messages can only be sent to the receiver system determined in the previous step.
9.3.2 Routing the Split Messages to Different Receiver Systems

To complete our discussion of the advanced routing options, we will briefly explain how you can configure a message split where the split messages are routed to different receiver systems. To do this, you define the message split using several 1:1 mappings (instead of one 1:n mapping).

This is the procedure:

1. In the ES Repository, define the necessary 1:1 mappings for each of the intended split messages. Each mapping creates another subset of line items out of the large source message.

2. In the Integration Directory, do the following:
   a) Configure the different receiver systems in a receiver determination.
   b) For each configured receiver, define an interface determination. In the interface determination, assign the right 1:1 mapping and inbound interface.

The behavior at runtime is illustrated in the figure below:

Figure 63: Configuring a message split with different receiver systems
10 B2B Integration

In the preceding chapters (in particular Designing Integration Content and Configuring Integration Content) we explained how to set up the basic communication with SAP NetWeaver PI. In particular, when we explained the configuration concepts, we assumed that all the details of a system landscape are known to the expert performing the configuration tasks. This is typically the case in small or midsize companies with a manageable size and structure of the system landscape. However, in larger enterprises or in scenarios spanning different enterprises, this assumption can no longer be made. In this section, we explain how business-to-business (B2B) scenarios can be managed with SAP NetWeaver PI. In a B2B scenario, business partners (this can be whole enterprises which are in a business relationship with each other) communicate and exchange data with each other based on an IT infrastructure without knowing all details of the whole system landscape.

In light of this fact, the concept behind the Integration Directory can be adapted to be used in a more flexible and generic way in order to cater for just such cases.


10.1 B2B Integration Design

In an integration scenario model, B2B communication can be represented as follows:

![Image of B2B Integration Scenario Model]

*Figure 64: Integration scenario model including a B2B interaction with the process component of an external business partner (grey)*

The process components interaction model showing the interaction with the external process component could look like this:
Figure 65: Process components interaction model showing a B2B interaction with the process component of an external business partner

SAP provides predefined integration content where B2B communication is anticipated in integration scenarios or process components interaction models like the one shown above.

### 10.2 B2B Configuration

When we explained the basic concepts for process integration configuration, we put a communication component, that is, an entity which could act as sender or receiver of messages, on the same level as a system (as described in the System Landscape Directory).

This corresponds to a situation where communication is being described for a system landscape that is known to the integration expert who performs the configuration tasks. This situation can be assumed as long as the integration only involves business applications that span parts of one and the same company. As soon as different companies share the same business process (in other words, in a B2B scenario), additional considerations come into play. Typically, in a B2B scenario, the configuration of the integration is a task that is distributed among integration experts from the involved companies or organizations. Each integration expert will only configure one “side” of the communication, and, in doing so, each expert knows only “his” part of the system landscape.

The next figure illustrates this fact schematically. It is a modified version of figure 3, including communication with an external business partner (business partner 2); in this case, the integration expert at business partner 1 does not know any details of the part of the system landscape hosted by business partner 2:
Interaction of Process Components 1 and 2

<table>
<thead>
<tr>
<th>Business Partner 1</th>
<th>Business Partner 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Systems Involved in Interaction 1 → 2

- Integration Broker
- Firewall

Figure 66: Modified version of figure 3, including B2B communication

The integration expert at business partner 1 only knows the part of the system landscape that is hosted by business partner 1. The complementary part of the system landscape hosted by business partner 2 is typically not known to him, since companies and organizations do not usually expose internal system names or server addresses to external partners. This part of the system landscape is a “black box” to him.

For an integration expert working for business partner 2, the reverse is true (assuming for simplicity that only two business partners share the business process).

There are additional configuration concepts to handle these B2B-specific constraints (implemented as object types or object attributes in the Integration Directory).

10.2.1 Communication Party

Since B2B scenarios typically involve whole enterprises interacting with each other, you use a communication party to identify a company or an organization that takes part in the business process. The communication party is an optional key field for receiver determinations, interface determinations, sender and receiver agreements. A party groups together those communication components that belong to the corresponding company or organization.

To accommodate the fact that B2B integration spans areas of responsibility that are separated from each other (as illustrated in the figure above), an additional concept is implemented: where you use an internal name to identify a company or a business partner during configuration, you have the option to map this internal name to a unique globally-recognized identifier that identifies the company unambiguously (for example, a D&B D-U-N-S number issued by the agency Dun & Bradstreet). In every communication with an external partner, the internal party name is then transformed to the globally-recognized identifier. Conversely, when a message is received from an external business partner, the identifier can first be mapped to the internal party name during inbound processing.
10.2.2 Business Component

In external communications (B2B interactions), business system names cannot be used as addressing entities since they are not exposed externally. Therefore, communication components based on business systems in the SLD are not suitable. Instead, messages involved in a B2B interaction use a different communication component type, called a business component. A business component merely represents an abstract entity for addressing the senders and receivers of messages in B2B communications.

10.2.3 Masking Internal Details in Outbound Messages

As we have seen in the section Configuring Integration Content, a message sent out to a receiver system in a company-internal interaction carries the information of the sender (the name of the sender system) in the message header. In an external or B2B communication, the internal system names should be hidden and not show up in the message header when the message arrives at the external business partner.

To make sure that internal system names are masked in message headers, a header mapping can be applied. With a header mapping, you define a transformation from the internal (sender) system name to an externally exposed business component name. You configure the header mapping in a receiver agreement (in other words: when you specify the outbound processing).

The following figure illustrates how header mapping works:

![Header Mapping Diagram]

Figure 67: Masking internal system names in an outbound message targeted at an external business partner

10.2.4 Routing Messages from an External Business Partner to an Internal System

If we consider the other direction of B2B message exchange, a message sent by an external business partner (2) cannot be addressed directly to a system of the internal landscape hosted by business partner 1. Instead, the external business partner addresses the message to a virtual receiver by using a business component name to address the receiver of the message. This business component name is the name you (business partner 1) have communicated to the external business partner beforehand. To route the message from the external business partner to the right system in your (business partner 1) internal landscape (whose name the external business partner does not know), you use a receiver-dependent receiver determination.

A receiver-dependent receiver determination is illustrated in the following figure:
10.2.5 Using the Model Configurator for B2B Integration

As far as you have already designed a process model in the ES Repository, you can also use the model configurator to automate the configuration.

In this case, those entities which have been designed as B2B-specific will also be handled accordingly by the model configurator. For example, for a process component that is labeled as process component at business partner in a process components interaction model, you can only assign a communication party (rather than a communication component based on a business system).


10.2.6 Additional B2B-Specific Configuration Settings

You can restrict the access of sender communication components of type business component or business system to particular service users.

For more information, see Restricting Access to Runtime Environment to Specific (Service) Users.

10.3 B2B Integration Based on Industry Standards

To address B2B integration challenges for specific industry sectors, several standards have been developed for the exchange of business data in these industries.

10.3.1 SAP Business Packages for RosettaNet and CIDX

SAP provides industry-specific business packages to support the B2B integration of industry standards, such as RosettaNet for the high tech industry.

Note: The RosettaNet Implementation Framework (RNIF) standard, developed by the organization RosettaNet, defines processes, interfaces, and transport protocols for the exchange of business data in the High Tech industry (for example in the electronic industry).
To facilitate the setup of B2B interactions with business partners compliant with an industry standard, SAP provides business packages containing both the relevant integration content in the ES Repository that allows the mapping to these standards, and the industry-specific adapters that enable an external standard to be technically connected with the integration broker. SAP provides business packages for RosettaNet, Chemical Industry Data Exchange (CIDX), and Standards for Technology in Automotive Retail (STAR).

10.3.2 Support of Electronic Data Interchange (EDI)

SAP NetWeaver Process Integration also supports the integration with United Nations/Electronic Data Interchange For Administration, Commerce, and Transport (UN/EDIFACT), the international standard for Electronic Data Interchange (EDI).

To ensure connectivity with systems based on this standard, you can use the SEEBURGER business packages for SAP NetWeaver PI. Over the past years, SEEBURGER has developed a large number of EDI adapters including thousands of mappings and dozens of EDI communication protocols, for different scenarios in various industries.

Customer Experience

SAP NetWeaver PI is widely used to integrate existing EDI landscapes. For example, a large consumer products supplier with 45,000 employees uses SAP NetWeaver PI in conjunction with SEEBURGER EDI adapters to integrate 4 different EDI system solutions in the enterprise, covering 2.5 million sales EDI transactions per year. Another example is Danfoss, an international leading manufacturer of valves and fluid handling components for heating, ventilating, and air conditioning, with about 21,200 employees worldwide (see Danfoss customer success story).

10.3.3 Structure of SAP Business Packages

SAP business packages are structured in the following way:

They contain:

- Integration content that describes the industry standard interfaces
- Integration content that describes the mappings of SAP standard interfaces to industry standard interfaces

For more information about structuring business packages and how the current industry standard is mapped as content in the Enterprise Services Repository, see Structure of SAP Business Packages.

- Adapter metadata for the industry standard adapter

You use the RNIF or CIDX adapter to connect to the message protocol supported by the relevant industry standard (RNIF adapter to connect to RosettaNet Implementation Framework and CIDX to connect to Chem eStandards).

You can use the business packages provided by SAP to define your own processes based on these industry standards. Usually, the most important task is to develop new mappings between your application and the interface of an industry standard.

More information: SAP Business Packages

10.3.4 Setting Up Integration Based on SAP Business Packages

You set up the integration by following the following general procedure:

1. Install the SAP business package

   Download the relevant SAP business package and import the corresponding industry-specific integration content to the ES Repository.

ESR Content (XI Content). Note that the authorizations for downloading process integration content are generated automatically depending on your licenses.

For example, look for the package XI CONTENT CIDX ERP for the integration content that describes the mappings of SAP standard interfaces to CIDX interfaces.

2. Define a new software component version in the System Landscape Directory for your development
   This software component version must be based on the software component version that contains the business package.
   See the explanations as provided under Using Predefined Integration Content.

3. Import the software component version to the Enterprise Services Repository
4. Define the content in the Enterprise Services Repository
   In this phase, you build your integration content based on the predefined industry-specific integration content you have imported.

   Note: The integration design for RosettaNet and CIDX is based on “classical” process integration scenarios.

5. Configure the integration
   We do not recommend using the model configurator to configure the integration. As the industry-specific content is designed based on process integration scenarios, make sure that you select the right process integration scenario for the industry-specific interface from the ES Repository.
   Consider the B2B-specific configuration concepts as described in this chapter and under Configuring B2B Processes.
   For more information, read the configuration guide for the business package. You can find the guide in SAP Software Distribution Center at http://service.sap.com/swdc on the info page for the corresponding download package.

Customer Experience

Advanced Micro Devices, Inc. (AMD) is an American multinational semiconductor company that uses SAP NetWeaver PI 7.1 and benefits from out-of-the-box process templates based on SAP business package for RosettaNet.
More information: Customer Scenarios with SAP NetWeaver PI 7.1
11 Security

In the preceding chapters, we focused on the concepts and tasks relevant to setting up the basic communication between the involved communication components. However, secure communication is a must for most productive scenarios, particularly in B2B scenarios.

In this chapter, we provide an overview of the basic security settings that are supported with SAP NetWeaver PI.

With SAP NetWeaver PI, you can apply security on the following levels:

- Ensuring a secure and reliable setup of the technical SAP NetWeaver PI landscape
- Ensuring a secure and reliable messaging when executing integration scenarios based on an SAP NetWeaver PI installation

As of SAP NetWeaver PI 7.1, the support of the standard Web Services Reliable Messaging (WS-RM) is considerably enhanced. We focus on this in the following chapter, but this chapter summarizes the most important security-related aspects.


11.1 Secure and Reliable Technical Landscape of SAP NetWeaver PI

An installation of SAP NetWeaver PI contains many technical components that communicate with each other when you operate an SAP NetWeaver PI landscape.

Note: When we talk about operating an SAP NetWeaver PI landscape, we mean the operation of all its technical components, for example the Integration Server, as well as the design and configuration tools.

The configuration of the technical components as well as the communication between the components has to comply with security standards. These technical components are, for example, messaging components like the Integration Server or the Advanced Adapter Engine, or tools for design and configuration like the ES Repository or the Integration Directory.


More information: Technical Communication

11.2 Secure and Reliable Messaging

When you actually run and integrate a collaborative process based on an SAP NetWeaver PI installation, the integration of the process components has to comply with certain security standards. In particular, when messages are exchanged between different enterprises or organizations over the Internet, it must be ensured that the message transfer is secure. The level of security depends on the kind of scenario. For example, B2B interactions that span different enterprises typically require higher security standards than interactions between components running within the same enterprise.

We basically distinguish between the two following security levels:

- Transport-level security

  Security on this level covers all security aspects of the technical communication level, particularly the technical communication protocols used. As a basic security setting for the technical connection, you can apply a specific transport security. The transport security you can apply depends on the adapter type used. For example, HTTP connections can be secured by using Secure Sockets Layer (SSL), resulting in the protocol HTTPS. When you use the IDoc or RFC adapter, you can apply the transport security Secure Network Communications (SNC).
Depending on the transport security, you can apply certain *authentication mechanisms* on the transport level. These are authentication mechanisms such as the use of users/passwords, client certificates, or SAP assertion tickets.

**Note:** Keep in mind that we mainly deal with system-to-system interactions. Therefore, users in this context are typically service users for logging on to a system.

You configure these settings in the communication channel for the corresponding adapter.

An additional option for increasing communication security is to build up several network zones by introducing firewalls and application gateways.

More information: Network and Communication Security

- **Message-level security**
  Security on this level covers all security aspects with regard to protecting the *content of a message* against malicious use. For example, you can digitally sign or encrypt documents exchanged between business partners.

  SAP NetWeaver PI supports message-level security for the XI message protocol, for the RosettaNet protocol, for the CIDX protocol, for the SOAP adapter, for the Mail adapter, and for connectivity with WSRM-enabled systems (based on Web Services Reliable Messaging).

  To configure message-level security settings, you have to perform the following tasks:
  - Security configuration in the connected back-end systems
    These settings include, for example, the setup of certificate stores and trust relationships between systems (using AS Java keystore or Trust Manager of AS ABAP).
  - Security configuration for the communication step in the Integration Directory
    These settings are made in the communication channel.

More information: Message-Level Security

You can find detailed information on the supported security settings for the various adapter types in the description of the adapter under Adapters.

It is not possible to provide general recommendations as to which security settings are the best. When you configure scenarios using interactions with WSRM-enabled systems (see Web Services Reliable Messaging), the Recommended WS Security Scenarios serve as a good starting point for planning your security strategy.

The actual configuration procedure depends on the protocol used and the applied security scenario.

### 11.3 Security Standards for Web Services

SAP NetWeaver PI (as of release SAP NetWeaver PI 7.1) supports several standards for Web services. We will explain how SAP NetWeaver PI supports these standards in the next chapter.

In this section, we focus on the security-relevant standards for Web services. These are the most important standards:

- **Web Services (WS) Security** (WS Security)
  A standard for securing SOAP messages.
  By using WS Security, you protect the SOAP messages that are exchanged between the Web service provider and the Web service consumer with digital XML signatures, XML encryption, time stamps, and security tokens.

- **Security Assertion Markup Language (SAML)**
  An XML-based language that can be used to exchange security-relevant information such as authentication information, for example.
SAML uses WS Security and is built upon a number of existing standards:

- **Extensible Markup Language (XML)**
  Most SAML exchanges are expressed in a standardized dialect of XML, which is the root of the name SAML (Security Assertion Markup Language).

- **XML Schema**
  SAML assertions and protocols are specified (in part) using XML Schema.

- **XML Signature**
  Both SAML 1.1 and SAML 2.0 use digital signatures (based on the XML Signature standard) for authentication and message integrity.

- **XML Encryption**
  Using XML Encryption, SAML 2.0 provides elements for encrypted name identifiers, encrypted attributes, and encrypted assertions (SAML 1.1 does not have encryption capabilities).

- **Hypertext Transfer Protocol (HTTP)**
  SAML relies heavily on HTTP as its communications protocol.

- **SOAP**
  SAML specifies the use of SOAP, specifically SOAP 1.1.

More information: Security Guide Web Services

As of SAP NetWeaver 7.1, the use of SAML tokens is possible.

### 11.3.1 Principal Propagation Using SAML 1.0

One use case for Web services security standards is principal propagation based on Security Assertion Markup Language (SAML). You can use principal propagation to pass the security credentials of a user from a sender to a receiver application (even spanning several intermediate components).

This security setting is supported for interactions with WSRM-enabled systems.

For end-to-end descriptions of the configuration steps, see the following sources:

- **How To Configure SAML Authentication for SAP NetWeaver Process Integration 7.1** (how-to guide)
- **Principal Propagation with SAP NetWeaver Process Integration 7.1** (SCN blog)

Note: For interactions based on the XI adapter, the RFC adapter, and the SOAP adapter, you can configure principal propagation based on SAP assertion ticket, an SAP-specific logon ticket.

More information: Configuring Principal Propagation (Authentication Assertion Ticket)

### 11.4 Restricting Access to Runtime Environment to Specific (Service) Users

You can restrict the access of sender communication components of type *business component* or *business system* to particular service users. An authorization check at runtime ensures that messages that have the particular communication component entered as the sender in the message header can only be executed on the Integration Server or in the Advanced Adapter Engine by the specified users.

Note: This security feature plays a role for B2B interactions in particular. You and your external business partner agree on a special service user to be used for communication.

More information: Access Control Using Assigned Users
11.5 Configuration Examples for Back-End Settings

The setup of end-to-end security mechanisms depends largely on the connectivity used, the security level, as well as the type of application systems connected to each other.

In this handbook, we mainly focus on the settings necessary to enable interactions between systems, mainly mediated by an integration broker. These settings – as already mentioned – are configured in the corresponding sender/receiver adapters and sender/receiver agreements. However, to make these settings effective, additional settings have to be configured in the connected application (back-end) systems. For procedure examples for configuring the back-end-specific settings, see the following configuration examples:

- Configuration Examples for AS ABAP
- Configuration Examples for AS Java

Note: These examples cover direct interactions between two systems – a service consumer and a service provider. For scenarios using an integration broker (for example, Integration Server), the integration broker has to be brought into consideration as a component interconnected between the consumer and provider application. From the perspective of the consumer system, the integration broker has to be treated as a service provider, whereas from the perspective of the provider system, the integration broker has to be treated as a service consumer.

For an overview of Web service security settings and procedures, see the guide Configuring Web Service Scenarios.

12 Services Registry

As explained in the chapter Designing Integration Content, a service interface is an implementation-independent description of a function or a service. In this sense, a service interface in the ES Repository (or its WSDL representation) represents only the metadata of a service that is already known at design time. However, to call, invoke, or consume a service, information about the location of the service is necessary, for example, the address and host name of the server the service is implemented on.

The Services Registry is a central registry in a landscape that contains service metadata enriched with the location information.

Note: The Services Registry is available together with SAP NetWeaver PI 7.1 and SAP NetWeaver Composition Environment 7.1.

In this chapter, we introduce the Services Registry, its basic concepts, and use cases.

More information: Publishing Service Definitions Using the Services Registry

12.1 Introduction

An important aspect of service-oriented architecture (SOA) is the facilitation of service consumption, either where the service is invoked by a mediated communication step, or directly by a consumer application (point-to-point communication). To make the invocation of services more convenient, the Services Registry was introduced. It is a UDDI 3.0-conformant registry containing information about the services provided in a landscape, with references to the relevant metadata (WSDL) and to the callable service endpoints.

A service endpoint contains a single configuration for a service. An endpoint contains the server address (location of the service) and additional security settings, for example, authentication methods to call the service. Multiple endpoints can be defined for one service, each endpoint representing a different runtime behavior.

The content of this chapter is structured according to the following basic use cases of the Services Registry:

- Service providers can publish their service information to the Services Registry and thus make the service accessible to any consumer.
Service consumers can browse the Services Registry for the service they need and discover and consume the service.

12.2 Publishing Service Definitions into the Services Registry

A service definition in the Services Registry can describe a service on different levels. It can represent a service interface description from the ES Repository, in which case the service definition contains the metadata of a service already known at design time. Beyond that, a service definition can represent an implemented service on a provider system. The third option is a configured service: an implemented service with an endpoint.

Taking into consideration mediated communication using an integration broker, a service definition can also represent the configured inbound processing on the integration broker.

Reflecting the different kinds or levels of service definitions, service providers can publish the following kinds of service definitions into the Services Registry:

1. Inbound service interfaces from the ES Repository
2. Service endpoints to the Integration Server/local Advanced Adapter Engine
3. Service definitions (AS ABAP) – implemented and configured
4. Service definitions (AS Java) – implemented and configured

Depending on the publication source, the service definition in the Services Registry is categorized by the attribute State as:

- Modeled – for service definitions containing only the service metadata from the ES Repository
- Activated/Deployed – for service definitions that are implemented, but not yet configured
- Configured – for service definitions containing endpoint information

The following figure illustrates how these different kinds of service publications relate to each other:
12.2.1 Publishing Inbound Service Interfaces from the ES Repository

In this option, the published service definition contains the description of the inbound service interface without any endpoint information (a “modeled” service). To call the service, the endpoint information has to be added later on.

More information: Publishing Service Interfaces

12.2.2 Publishing Sender Agreements or Integrated Configurations from the Integration Directory

If you have configured mediated communication, you can regard the sender application as a service consumer that calls a service provider (the receiver application) using the integration broker that is interconnected between consumer and provider. The integration broker can either be the Integration Server or the local Advanced Adapter Engine. However, from the consumer’s perspective, the integration broker acts as an “intermediate” service provider, and the configured inbound message processing implicitly contains the service endpoint information for calling the integration broker.

To make it easy for any consumer to call the integration broker (based on a specific configuration), you have the option to publish the configured inbound processing into the Services Registry. Depending on whether you have configured Integration Server-based messaging or local messaging using the Advanced Adapter Engine, you have the option to publish either of the following Integration Directory objects:

- Sender agreements (with WS channels or SOAP channels assigned to them) – for Integration Server-based communication
- Integrated configurations (with SOAP channels assigned to them) - for local message processing based on the Advanced Adapter Engine
Note: The system information for published sender agreements or integrated configurations contains the addition \_XPI \.

The following figure illustrates this use case:

![Diagram illustrating the process of publishing a sender agreement or integrated configuration into the Services Registry](image-url)

**Figure 70: Publishing a sender agreement/integrated configuration into the Services Registry**

More information:

- [Publishing Sender Agreements into the Services Registry](#)
- [Publishing Integrated Configurations to the Services Registry](#)

### 12.2.3 Publishing Service Definitions (AS ABAP or AS Java)

You can publish a configured service implemented on a back-end provider system directly into the Services Registry. Using this option, you facilitate the consumption of the provider service for any consumers.

More information:

- [Publishing Services (ABAP)](#)
- [Publishing to the Services Registry (Java)](#)

### 12.3 Discovering Services in the Services Registry

Whereas service providers use the Services Registry as a registry to publish the services they offer, service consumers use the Services Registry to search for and discover the services they require.

#### 12.3.1 Service Consumption with the Services Registry

More information: [Service Consumption with the Services Registry](#)

#### 12.3.2 Browsing and Searching for Services

Service consumers can search in the Services Registry for services or service groups.

More information:
Searching and Browsing Service Definitions
Searching and Browsing Service Groups

12.3.3 Classifying Services

To structure the data accessible in the Services Registry, service definitions can be categorized based on a classification system. SAP provides a set of classification systems which allow you to group services, for example, according to ES Repository metadata, like software component, process component, or service interface.

More information:
Structuring of Services in the Registry
Classifying Services
Managing Classification Systems
How-to guide: Services Registry: Classification Services

12.4 Additional Tasks

For more information on additional tasks to perform with the Services Registry, read the following documentation:

- Testing Service Endpoints
- Importing or Exporting Services Registry Content
- Cleaning the Services Registry
- Removing Services from the Services Registry
- Using the Services Registry APIs

12.5 SAP Enterprise Services Explorer tool for Microsoft .NET

SAP Enterprise Services Explorer for Microsoft .NET is an add-in for Microsoft Visual Studio 2005 and 2008 that enables .NET developers to discover services provided by SAP and consume them in their applications, as well as publish their own .NET web services.

More information:
Introducing SAP Enterprise Services Explorer tool for Microsoft .NET
Downloading SAP Enterprise Services Explorer tool for Microsoft .NET

13 Web Services Reliable Messaging

The Web Services Reliable Messaging (WSRM) specification issued by OASIS in 2005 describes a protocol that allows messages to be transferred reliably between nodes implementing this protocol in the presence of software component, system, or network failures.

A WSRM-enabled system guarantees a certain transactional behavior that ensures that a consistent data state is achieved after the interaction, even in the event of failures as mentioned above.

More information: Web Services Reliable Messaging

SAP NetWeaver Process Integration (as of release SAP NetWeaver PI 7.1) supports WSRM with the following enhancements:

- Connectivity
  It is possible to connect WSRM-enabled systems to the Integration Server and to configure certain security settings.
In addition to this, you can also configure the direct (point-to-point) connection between WSRM-enabled systems.

You can configure a communication channel with adapter type WS for this purpose in the Integration Directory.

- Interface design

In the ES Repository, you can define certain interface patterns for service interfaces that help to implement WSRM-enabled applications. These are the interface patterns Stateless and TU&C/C.

Note: For example, a TU&C/C scenario (Tentative Update and Compensate or Confirm) might be designed as follows: A service consumer sends his orders to a service provider. The provider processes the orders tentatively. Only after an order is confirmed on the consumer side is the order also persisted in the database on the provider side. In the event of an error, the changes are rolled back. To implement such behavior, multiple operations have to be designed for the service interface and implemented later in the related application systems (more information: Interface Patterns).

In this chapter, we provide an overview of the configuration options for connecting to WSRM-enabled systems. Basically, you have the following options:

- Connecting the Integration Server to Web service consumers and providers
- Setting up direct communication between Web service consumers and providers

Note: If it is necessary to distinguish between the roles of a service provider and a service consumer, in the context of WSRM-enabled systems we also speak of Web service consumer (WS consumer) and Web service provider (WS provider).

The end-to-end configuration process for scenarios includes both tasks in the Integration Directory and tasks in the connected back-end systems, and depends strongly on the kind of back-end system.

You can find detailed information on the configuration of scenarios including Web service consumers and providers for different combinations of back-end system types in the following guide: Configuring Web Service Scenarios.

13.1 Connecting the Integration Server to WSRM-Enabled Systems

As of SAP NetWeaver PI 7.1, you can connect WSRM-enabled systems to the Integration Server. The configuration procedure is basically the same as described under Configuring Integration Server-based Communication and Configuring Communication Using the Integration Server.

The basic difference is that you use a communication channel of adapter type WS to configure the connectivity to the WSRM-enabled system:

- You use a sender agreement with an assigned WS sender adapter to connect the Integration Server to a WS consumer.
- You use a receiver agreement with an assigned WS receiver adapter to connect the Integration Server to a WS provider.

In particular, in the agreements and channels you can configure WSRM-specific security settings such as authentication mechanisms.

More information: Configuring the Communication Channel with Adapter Type WS

13.2 Setting Up Direct Communication between WSRM-Enabled Systems

To configure direct communication between WSRM-enabled systems, you configure the relevant settings locally in the involved back-end systems of the WS provider and WS consumer.

- For systems based on AS ABAP, you generally use the SOA Manager.
For systems based on AS Java, you generally use the SAP NetWeaver Administrator. The different procedures for different kinds of back-end systems are described under Configuring Web Service Scenarios.

If the WS provider and consumer are both based on AS ABAP 7.1 or a higher release, you can perform the configuration of the direct communication centrally in the Integration Directory, using a direct connection object (with an assigned sender channel of type WS).

The configuration settings are propagated into the back-end systems by a caching mechanism, therefore making local configuration unnecessary.

More information: Configuring Direct Communication
14 Appendix

14.1 SAP NetWeaver PI Architecture and Tools

14.1.1 Technical Components of SAP NetWeaver PI

The following figure provides an overview of the technical components of SAP NetWeaver PI:

![Figure 71: Technical components of SAP NetWeaver PI](image)

More information: Technical System Landscape

14.2 Service Provisioning

Service provisioning means making a service available to be called by any service consumer. So far in this handbook we have not focused on this aspect of SOA implementation. Rather, we focused on how to integrate service providers and service consumers – mainly using an integration broker. However, service provisioning deals with all tasks that are related to making a service available to any kind of service consumer, regardless of whether the service consumer calls the provider directly or mediated by an integration broker. In the latter case, the integration broker acts as a service provider from the perspective of the service consumer.

Although we have not explicitly covered the topic of service provisioning, you can derive the necessary concepts and tasks from the content of this handbook. The topic of service provisioning covers a subset of the topics, dealing in particular with these integration project tasks:

1. Designing interface objects
2. Proxy generation
3. Publishing service endpoints into the Services Registry

The subsequent step, the service discovery, is not covered in this handbook.

For more information, see:

Service Enabling with SAP NetWeaver Process Integration 7.1

### 14.3 Setting Up High-Volume Scenarios

There are several methods for increasing the performance of message processing and enabling the processing of very large messages.

This chapter provides a short overview of the possible methods and provides references to more detailed information.

More information: SAP NetWeaver PI Best Practices: Sizing & Performance Tuning (this how-to guide provides best practices related to SAP NetWeaver PI sizing and performance improvement)

#### 14.3.1 Local Message Processing Using the Advanced Adapter Engine

Processing messages locally using the Advanced Adapter Engine can increase performance considerably, since the Integration Server is bypassed. However, other decision criteria must be taken into account, since the mediation capabilities of local message processing are not congruent with those offered by the Integration Server.

More information on how to configure such scenarios:

Configuring Communication Using the Advanced Adapter Engine
Section Configuring Local Message Processing on the Advanced Adapter Engine

#### 14.3.2 Message Packaging

You use message packaging to group together asynchronous messages in packages and then process each message package in one logical unit of work.

More information:

Message Packaging
SAP Note 1037176

### Customer Experience

Customers benefit from performance enhancements for high volume scenarios when upgrading to SAP NetWeaver PI 7.1. For example, Swiss Post, a Switzerland-based logistics company that deals with postal, logistics, and transportation services (around 58,000 employees) uses SAP NetWeaver PI 7.1 to implement business-critical high-volume processes. These are some performance numbers from Swiss Post: about 1.2 million messages are processed overnight; response times of 200ms for an end-to-end B2B interaction have been measured.

Energie Baden-Württemberg AG (EnBW), the third-largest energy company in Germany, benefits from performance improvements by using local processing on the Advanced Adapter Engine and the message packaging functionality. Another SAP NetWeaver PI 7.1 customer benefitting from the performance improvements in the new release is Swisscom IT Services AG.

More information:

Customer Scenarios with SAP NetWeaver PI 7.1
High Volume Messaging with SAP NetWeaver PI - Real Life Customer Examples
14.4 Additional Connectivity Options

14.4.1 Using SAP Conversion Agent by Informatica

An additional connectivity option is provided by SAP Conversion Agent by Informatica. This tool, developed in partnership with Informatica, enables efficient development of data transformations for incoming and outgoing unstructured and semi-structured data and messages into XML.

More information: [SAP Conversion Agent by Informatica](#)
14.5 Glossary

adapter
Defines the technical transformation of a message necessary to connect a system or another kind of communication partner to the integration broker. A sender adapter transforms the message from the format a sender system uses into the message format of the integration broker. A receiver adapter transforms the message from the message format of the integration broker into the format a receiver system can handle. Adapters are configured in the Integration Directory using a communication channel.

aggregated data type
Comprises other aggregated data types or core data types or both.

asynchronous communication
Communication mode where a request message is sent out but no response is expected.

business-to-business (B2B) communication
Communication with an external business partner.

business system
Entity in the System Landscape Directory that represents a logical system. Each business system must be assigned a technical system.

category
Attribute of a service interface that determines whether the service interface is an outbound, an inbound, or an abstract service interface.

communication channel
Configuration object that is used to configure an adapter.

communication party
Represents a company unit that is to be involved and addressed in message exchange.

communication component
Represents an entity that can be used to address a sender or receiver of messages (typically, a system).

communication channel template
Contains the preconfiguration of a communication channel that can be specified at design time in the ES Repository.

communication mode
Attribute of a service interface operation that determines whether the communication has to be performed synchronously or asynchronously.

configuration object
Object created and maintained at configuration time in the Integration Directory.

configuration time
Phase of an integration project during which the integration content that has been created at design time is configured to run in a specific system landscape.

connectivity
Capability to connect systems or business partners based on different protocols or standards to the integration broker.

content-based routing
Routing that depends on the content of a message.

**context object**
Design object that can be used as an abbreviated expression for an XPath expression to address a specific payload element.

**core data type**
XSD-based type that does not yet have any business-specific semantics.

**data type**
Design object that describes the structure of data. Data types contain elements and properties for the type definition. Data types are saved as XML Schema Definition.

**deployment unit**
Groups all process components that need to be installed together on one system.

**design object**
Object created and maintained at design time in the ES Repository.

**design time**
Phase of an integration project during which those parts of a business process are specified that are independent of any implementation- or system-landscape-relevant technical details. The content created and maintained at design time is referred to as integration content.

**direct communication**
Communication type where systems interact directly with each other without an integration broker interconnected between them. This kind of communication is also referred to as “point-to-point” communication.

**direct connection**
Configuration object used to configure direct communication between a sender and receiver system.

**enterprise service**
Term used to emphasize the fact that a service has been designed according to SAP’s SOA design principles. Technically, this term summarizes interface objects.

**Enterprise Services Repository (ES Repository)**
Central repository within SAP NetWeaver for design objects for service development and their metadata.

**Enterprise Services Workplace (ES Workplace)**
The ES Workplace is the central place to view consolidated information about all available enterprise services delivered by SAP. You can access the ES Workplace in SAP Community Network at [https://www.sdn.sap.com/irj/sdn/explore-es](https://www.sdn.sap.com/irj/sdn/explore-es).

**external definition**
Object used to import an existing data structure description into the ES Repository.

**fault message type**
Message type used to describe the expected message if an error occurs.

**global data type (GDT)**
Data type that describes a basic business entity such as an address. GDTs are the basis for application-specific data types SAP-wide and can be used by customers to build their own data types.

**imported object**
XML definitions of existing functions (RFCs or IDocs) that can be imported into the ES Repository.
imported archive
Encapsulates an externally developed Java or XSLT (Java) mapping program. An imported archive has to be referred to by an operation mapping.

inbound processing
Defines how a message should be transformed technically to the XML message format of the integration broker.

inbound service interface
Service interface that receives a message.

integration broker
Term used to indicate the instance that processes messages in mediated scenarios.

The integration broker can either be the Integration Engine (as part of the Integration Server) or the local Advanced Adapter Engine.

integration content
Those parts of the Enterprise Services Repository content (ESR content) that are used for the design of the integration-specific aspects of business processes.

integration engine
Runtime environment based on AS ABAP that has the task of receiving and forwarding messages on the Integration Server or in SAP Business systems.

integration process
Executable cross-system process for processing messages.

In an integration process you define all the process steps to be executed and the parameters relevant for controlling the process. An integration process is a design object in the ES Repository.

interface determination
Configuration object used to specify the inbound interface that receives the message for a specific receiver system as well as to choose the mapping that has to be applied.

interface pattern
Describes the type of communication that is to be executed on the message when the interface is used.

integrated configuration
Configuration object for scenarios where messages are processed locally using the Advanced Adapter Engine.

integration content
As the design-time-relevant aspects are specified and stored in the Enterprise Services Repository (ES Repository), the corresponding design-time entities are referred to as integration content or (denoting the tool) ESR content. Therefore, we also speak of integration content predefined by SAP in this context.

integration scenario (integration scenario model)
Model type in the ES Repository that shows the graphical representation of the deployment units and process components involved in a cross-component business process, as well as the connections between the process components.

mapping template
Design object that contains parts of a message mapping that can be used as a copy template to create new message mappings. Mapping templates can reference other mapping templates, thus enabling maximum reuse in mapping development.
mediated communication
Communication between systems or other kinds of communication partners where a central instance is interconnected between the communication partners and acts as a communication hub or data hub. The data hub is referred to as integration broker.

message mapping
Mapping object created when you use the graphical mapping editor. A message mapping has to be referred to by an operation mapping.

message type
Defines the root element of a message. You use a message to exchange data between systems. A message type has exactly one data type that defines the structure of this data.

object key
Set of attributes that is necessary to identify an object (ES Repository and Integration Directory).

operation
Represents the smallest, separately-callable function, described by a set of data types used as input, output, and fault parameters serving as a signature. Operations are grouped by a service interface.

operation mapping
Mapping object that assigns a mapping program for a pair of service interface operations.
An operation mapping encapsulates the mapping program that has to be executed at runtime. At configuration time, you only have to choose the required operation mapping (in an interface determination or an integrated configuration); the assigned mapping program will then automatically be evaluated at runtime.

outbound processing
Defines how a message should be transformed technically from the message format of the integration broker into the format a receiver can process.

outbound service interface
Service interface by which a message is sent out to another process component.

process component
Part of the value chain of a business application or a business process. Assuming the business application spans different departments of one company, a process component typically represents one part of the process that is performed in one department.

process components interaction model
Model type in the ES Repository that shows in detail the interaction between two process components.

process component at business partner (external process component)
Process component that is connected by B2B communication with a process component in your own business and in this way enhances an intra-enterprise value chain.

process components interaction model
Shows in detail the interaction between two process components. You can consider a process components interaction model as a detailed view of one single connection in an integration scenario model. This model type is available as of SAP NetWeaver PI 7.1.

process integration scenario
Model type in the ES Repository that describes a cross-component process that is based on the exchange of messages. Process integration scenarios were already available in the earlier releases of SAP NetWeaver PI prior to SAP NetWeaver PI 7.1.
product
Represents a unit that is delivered, visible to the customer, and is installable and renewable. In an SAP environment, a product corresponds to an SAP technical component.

principal propagation
Passing the security credentials of a user from a sender to a receiver

proxy
Platform-specific, executable service derived from a service description in the ES Repository.

receiver determination
Configuration object that defines which receiver systems an incoming message has to be forwarded to by the integration broker.

receiver rule
Configuration object used to specify the conditions under which messages are forwarded to particular receivers. You can reuse a receiver rule in different receiver determinations.

routing
Defines which receiver systems and receiver interfaces a message is forwarded to by the integration broker. Routing is configured in the Integration Directory using receiver determinations and interface determinations or integrated configurations.

SAP NetWeaver Exchange Infrastructure
Name of SAP NetWeaver Process Integration in releases including and prior to SAP NetWeaver 2004.

SAP NetWeaver Process Integration (SAP NetWeaver PI)
SAP’s solution for the integration of applications, business partners, and services in complex and heterogeneous system landscapes. SAP NetWeaver PI provides an integrated tool infrastructure to model and design the integration-relevant aspects of business processes, to specify the integration-relevant details for a concrete system landscape, as well as to execute and monitor the business processes executed at runtime. In former releases, SAP NetWeaver PI also is referred to as usage type Process Integration of SAP NetWeaver 7.0, respectively SAP NetWeaver Exchange Infrastructure 3.0.

service
Coherent set of callable operations clearly described by one service interface, behavioral information, and endpoint information (such as the address where the service is provided). An operation is described by its input, output, and fault parameters, which in turn are defined by data types.

service endpoint
Contains a configuration for a service, including the service location and security settings.

service interface
Design object representing a set of functions that is either provided by an application (inbound service interface) or used by an application (outbound service interface).

software component
Represents the reusable modules of a product. You can upgrade them or install patches on them.

software component version
Version of a software component.

synchronous communication
Communication mode where a response is expected for each call or request message sent out.
target field mapping  
Describes how one or more source fields are mapped to a target field. A message mapping is constructed out of individual target field mappings.

technical system  
Entity in the System Landscape Directory that contains the technical details of a system (for example, the server address).

top-down approach  
Model-based design of integration content.

value mapping  
Transforms values of message elements.

value mapping group  
Configuration object in the Integration Directory that contains mappings between different representations of one and the same object. Mapping programs can refer to a value mapping group.

Web Services Description Language (WSDL)  
XML-based language for describing Web services and how to access them.

Web Services Reliable Messaging (WS-RM)  
This WS-ReliableMessaging specification describes a protocol that allows messages to be transferred reliably between nodes implementing this protocol in the presence of software component, system, or network failures. (OASIS definition)
14.6 Further Reading and Training

The SOA Middleware homepage in SAP Community Network is the first point of entry for up-to-date information about SAP NetWeaver Process Integration.

SAP Training and Learning Offerings

Training courses:

- BIT400: SAP NetWeaver Process Integration
- BIT469: SAP NetWeaver PI - Mapping
- BIT480: SAP NetWeaver PI – Operations
- BIT420: SAP NetWeaver PI - Proxy Development and Service Provisioning
- SOA100: SAP SOA Overview
- SOA110: SOA Roadmap and Methodology
- SOA300: Enterprise Service Repository
- SOA400: SOA Service Provisioning and Consumption

For more information, see the SAP training catalog.

Additional learning material:

- Learning Map SAP NetWeaver PI 7.1 (including EHP 1) (SAP Service Marketplace login required)

Process Integration and Service-Oriented Architecture (SOA)

SOA Development Handbook

SAP Press Books

SAP Exchange Infrastructure (currently being updated to SAP NetWeaver PI 7.1)

SAP Exchange Infrastructure for Developers (currently being updated for SAP NetWeaver PI 7.1)

B2B Integration Using SAP NetWeaver PI

Mastering SAP XI Administration

Mastering SAP NetWeaver XI - Programming

Enterprise Integration Patterns for SAP NetWeaver PI


Installation and First Steps

Install, Configure, and Run SAP NetWeaver PI 7.1

Master Guide and Installation Guides (SAP Service Marketplace login required)

SAP NetWeaver PI Configuration Guide

Use Cases and Examples

Typical use cases of SAP’s SOA Middleware (based on SAP NetWeaver PI):

- Service-Enabling a Legacy Application with SAP NetWeaver PI 7.1
- Service Enabling with SAP NetWeaver Process Integration 7.1

Step-by-step guides:

- Simple Use Cases (SAP NetWeaver PI 7.1)
- Demo Examples (SAP NetWeaver PI 7.1)
- Demo Examples Exercise Cancel Flight Booking

For more demo examples, see SAP Community Network ➔ Getting Started With Service Bus-Based Integration ➔ Use Cases and Examples.

SAP Library Documentation (SAP Help Portal)

**SAP NetWeaver Process Integration Library**

Functional units of SAP NetWeaver PI:

- Enterprise Services Repository
- Services Registry
- Integration Directory
- SAP NetWeaver Administrator
- SOA Manager
- System Landscape Directory
- Integration Engine
- Advanced Adapter Engine
- SAP Conversion Agent by Informatica
- Partner Connectivity Kit
- Adapter Engine (Java SE)

Operating SAP NetWeaver PI:

- Technical System Landscape provides an overview of the components of SAP NetWeaver PI
- Process Integration Troubleshooting Guide (SAP NetWeaver PI 7.1) (SMP login required)

Additional Sources

How SAP NetWeaver PI 7.1 is Used in Customer Landscapes

SAP NetWeaver PI 7.1 – Overview

SAP NetWeaver PI 7.1 - Detailed Overview

SAP Enhancement Package 1 for SAP NetWeaver PI 7.1

Enterprise Services Workplace

Enterprise Services Wiki