



PI Best Practices: Modeling

Applicable Releases:

SAP NetWeaver Process Integration 7.1x

SAP NetWeaver CE 7.1

SAP NetWeaver 7.x

Topic Area:

SOA Middleware

Capability:

SOA Management

Version 1.0

May 2009

© Copyright 2009 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice.

These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

These materials are provided "as is" without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall not be liable for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials.

SAP does not warrant the accuracy or completeness of the information, text, graphics, links or other items contained within these materials. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third party web pages nor provide any warranty whatsoever relating to third party web pages.

SAP NetWeaver "How-to" Guides are intended to simplify the product implementation. While specific product features and procedures typically are explained in a practical business context, it is not implied that those features and procedures are the only approach in solving a specific business problem using SAP NetWeaver. Should you wish to receive additional information, clarification or support, please refer to SAP Consulting.

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressly prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP's Support Services and may not be modified or altered in any way.

Document History

Document Version	Description
1.00	First official release of this guide

Typographic Conventions

Type Style	Description
<i>Example Text</i>	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, graphic titles, and table titles
Example text	File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
Example text	User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

Icons





Icon	Description
	Caution
	Note or Important
	Example
	Recommendation or Tip

Table of Contents

1.	Introduction.....	1
2.	Background Information.....	2
3.	Prerequisites	2
3.1	Prerequisites.....	2
3.2	Supported releases	2
3.3	Target Audience	2
3.4	Relevant Documentation	2
4.	Modeling Basics	3
4.1	Modeling Approaches.....	3
4.2	Define Enterprise Services	4
4.3	Modeling Entities	5
4.3.1	Business Object	5
4.3.2	Process Components.....	6
4.3.3	Deployment Units.....	7
4.3.4	Service Operations	8
4.3.5	Service Interfaces	10
4.4	Interface Patterns	11
4.4.1	Modeling A2A Interactions	11
4.4.2	Modeling A2X Services.....	14
4.5	Transaction Communication Patterns	16
5.	Modeling Services in ESR	18
5.1	Model Types in ESR.....	18
5.1.1	Process Component Models.....	18
5.1.2	Integration Scenario Models	19
5.1.3	Process Component Interaction Models.....	20
5.1.4	Business Object Map	22
5.1.5	Integration Scenario Catalogue	22
6.	Process Integration Scenarios.....	23
6.1	Application Components.....	23
6.2	Actions	23
6.3	Connections.....	23

1. Introduction

In a Service Oriented Architecture for business applications, you ideally model your application prior to implementation. The aim of modeling is to model business process flows. The model helps you to understand and then to implement the process flows or to enhance an implementation that already exists. SAP NetWeaver provides capabilities for modeling business processes at different levels of abstraction. Conceptual process modeling sustains large-scale business process analysis projects, which drive process harmonization and standardization and aim for a high degree of process excellence. Although this works very well for the stable core of the organization, the need to integrate with individuals, business partners and third-party systems, and above all to remain innovative, requires agile methods for supporting the translation of functional business requirements into the technical specifications of process execution. SAP NetWeaver supports unstructured collaborative workflows, as well as highly-structured integration processes for service orchestration. At the heart of Enterprise SOA, the Enterprise Services Repository exposes application core processes using modeled process components and enterprise services that act as key artifacts to compose new process innovation at the edge.

The modeling environment in SAP NetWeaver enables distributed business process modeling throughout the company and offers administration and analysis functions. The Enterprise Services Builder in the SAP NetWeaver PI offers a modeling environment where you can create various models in the ES Repository.

A model-driven service development provides the following advantages:

- A process should be part of a model-driven process whereby services of new applications are adapted in cooperation with other development departments.
- Depending on the modeling, you can work out which design objects are necessary in the ES Repository for your application.
- Interface patterns in the modeling ensure that services are always defined and named in the same way.
- Models are good way of documenting the whole process of an application and make it easier to enhance the software later.

There are two modeling environments in the Enterprise Services Repository:

- **Process Component Architecture Modeling:** This modeling environment enables SOA governance and the model-based design of service-enabled business applications. These models represent a formalized modeling approach as the modeling entities (process components, enterprise services, service operations, and global data types) are deployed based on the models in the application platform. Process Component Architecture Modeling supports process integration modeling using process components in the Enterprise Services Repository (delivered with SAP Net Weaver Composition Environment 7.1 and SAP NetWeaver Process Integration 7.1). The embedded modeling methodologies and service-based reference models help to understand the business semantics of enterprise services in the business process platform. The process component is the central modeling object and exposes the operations, service interfaces and business objects used. It references to several model types to combine the needed information in one place.

With Process Component Architecture Modeling the foundation for process composition is given. Customers can drill down from graphical high level models to service interfaces and operations. This empowers transparency of the SOA design. SAP works with this environment to create models for the applications in the SAP Business Suite. These models are shipped to customers. Customers can also create their own models in the ES Repository.

- **Process Integration Scenario Modeling:** This modeling environment in the ES Repository was part of previous SAP NetWeaver PI releases and continues to be supported in the latest release. It concentrates on the modeling of the exchange of messages between application components.

2. Background Information

This guide is part of a how-to guide series providing best practices and guidelines for PI and SOA processes. To increase the level of reusability, SAP works internally with the modeling environment, including the unification and governance process. SAP delivers the models that are produced by this process as ESR content to aid the understanding of SAP Applications. This Guide focuses on the principles of modeling and the various model types and interfaces. SAP recommends that customers conform to these principles when they create their own models.

3. Prerequisites

3.1 Prerequisites

- Basic understanding of SOA concepts.
- Working knowledge of SAP NetWeaver and SAP Business Suite.

3.2 Supported releases

SAP NetWeaver PI	7.10 and higher
SAP NetWeaver CE	710 >= SP03 with ESR installed
SAP NetWeaver	70 >= SP14

3.3 Target Audience

This document mainly focuses on the Architects, Developers, IT Managers, business process experts and business analysts who want to learn how to do services modeling and design with Enterprise Services Repository in SAP NetWeaver.

3.4 Relevant Documentation

[Enterprise SOA – Business Object Modeling Guideline – I](#)

[Enterprise SOA – Business Object Modeling Guideline - II](#)

[Enterprise SOA – Service Operation Design Guideline](#)

[Enterprise SOA – Global Data Type \(GDT\) Design Guideline](#)

[Enterprise SOA – Documentation and Naming Guideline](#)

4. Modeling Basics

Modeling capabilities empower enterprise service architects to have a holistic view of the analysis, design and architecture of service oriented assets. Enterprise service architects can now use service modeling disciplines to provide strategic and tactical solutions to enterprise problems. Through modeling, architects focus on design, ensuring reusability, naming conventions, and scalable interaction and integration scenarios. In the below sections, we will see the various concepts of the modeling and their best practices.

4.1 Modeling Approaches

Modeling is based on different approaches, each having its pros and cons. The approach used depends on whether the process modeler's objective is an overall strategic model or a detailed tactical model.

- **Top-down Approach:** The top-down approach focuses on the overall process. It starts by identifying the business processes and business services used by business users and there by defining the overall business requirements that give the framework of the business process model. Then the model that describes how a requirement is fulfilled is split into separate business processes. Each of these processes has its specific activities and the activities include specific services and tasks. The goal of the model, however, is to depict a broader view of a business process fulfilling a given requirement but not to show how a specific activity is performed. This broader view helps business analysts and managers to see how the overall process is going, where it needs improvements, and whether there are missing elements in the general process.
- **Bottom-up Approach:** In contrast to the top-down approach, the bottom-up approach starts by defining the activities at the base of the process model. Using them, many different and detailed business processes are created, to describe how low level business requirements are fulfilled. The goal of the model is to show how a specific activity is performed to produce a value at the end of the process. In other words, this approach focuses on the sub-processes first. This detailed view helps developers and system architects to make the process work. However, a problem occurs when all these details have to be combined to form the overall model picture, or when it comes to defining the key requirements that the general model should fulfill.
- **Inside-out Approach:** Unlike the top-down and bottom-up approaches, which are rather vertical types of modeling, the inside-out one is a horizontal approach. It consists of defining key processes in the overall process and then complementing them with other processes. This approach may be helpful to modelers in different areas if neither of the vertical approaches is appropriate. Like both other approaches, it also has disadvantages. One of them is that when using this approach there might be difficulties in defining what the key processes are.

Best Practice Guideline:

The best practice is to follow the **Top-down** approach because of the following reasons:

1. It provides granular functions which can be reused in various processes.
2. It provides the necessary flexibility for building processes on top of a stable platform.
3. This approach can be broken down in a phase model starting with the business requirements and ending with the implemented composite.

This recommended approach typically involves the following steps:

- a. Analyze business requirements

- b. Specify process information about the business processes and composite applications
- c. Think about exception handling.
- d. Determine business objects.
- e. Describe the user interfaces (interactive steps)
- f. Describe the required services.

4.2 Define Enterprise Services

The starting point of an enterprise service implementation is ES Repository. There are two approaches for defining new enterprise services.

- **Inside-out Approach:** When you already have functionality needed for your business purposes implemented in a backend application, you may only need to expose that functionality as web services, thus enabling your legacy application for SOA development. This approach of service enabling existing functionality is known as Inside-out development of services. In this approach, you expose an implementation that is already available as service; you use the service signature in the system to generate a service description and then publish it externally. By providing services in this way, you lose the benefits of governance, homogeneity, and so on.
- **Outside-in Approach:** In this approach, you start with modeling and move towards the implementation to create Enterprise services. It starts at business level, looking at critical business processes and modeling them into services that implement those processes. This approach is suitable if the functionality that is required does not already exist in the system. In this approach, you first model your service definitions and create design objects in the ES Repository independent of the language that will be used afterwards to implement the actual business logic of the operations.

Best Practice Guideline:

SAP's objective is to provide integrated solutions, which allow the execution of business processes beyond enterprise boundaries, between SAP and Non-SAP applications and components. The different applications and components must communicate with each other in a clear and consistent manner via Enterprise Services. This is ensured by the **Outside-in** approach when creating service interface operations. So the best practice is that to use the **Outside-in** approach when creating the new services. In this case, the development of the service consists of the following steps:

- Model your service in the Enterprise Service Repository:
 - Design the service interface and its operations.
 - Design the message type
 - Assign the data type to each message type
- Define your service in your development environment:
 - Generate the provider proxy for the service interface
 - Implement the provider proxy source code
 - Create a runtime configuration
- Publish your service in the Service Registry.

4.3 Modeling Entities

Service modeling in ES Repository is based on a set of predefined modeling entities and patterns that ensure that all services are always defined and named in the same way. Following these modeling principles and patterns provides the homogenous granularity of your services. These modeling entities are representations of logical business content such as a **Purchase Order** business object that is significant to the business. Figure1 illustrates the meta model of an enhanced enterprise service.

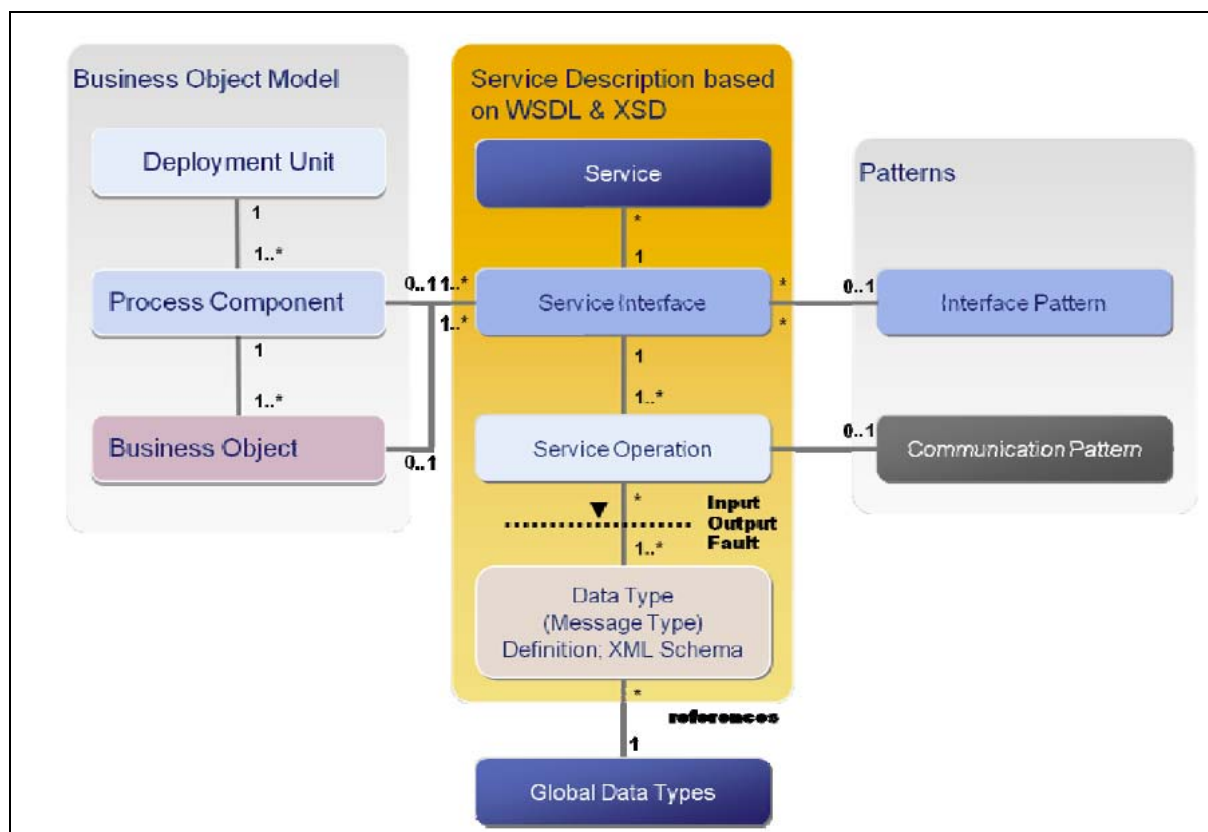


Figure 1: Meta model of an Enhanced Enterprise Service

These modeling entities for service definitions and their best practices are discussed in detail below.

4.3.1 Business Object

Business Objects describe the data of a well-defined and outlined business area. Business objects are defined free of business functionality redundancies and therefore serve as the central point and basis for modeling and defining services. Business objects represent a set of entities with common characteristics and common behaviors, which, in turn, represent well-defined business semantics. A **Sales Order** is an example of a business object.

Example:



Figure 2: Business Object

Best Practice Guideline:

- To be able to use Business Objects effectively for multiple use cases, you must model them in such a way that you avoid overlaps with other Business Objects. For example, product data should not be part of an order, but rather modeled in a separate Business Object. In this way the Business Object for product data can then be used for other use cases.
- You can also use the Business Object as a template for a whole series of Business Objects. For example, a business partner can be either a customer or a supplier and the data, with which either the customer or the supplier is described, is the same. If you consider beforehand a “business object template” from which the customer and supplier can be projected, then you avoid an inconsistent definition of the same data.

- Naming Rule:

Syntax: <qualifier>*<object>?

<qualifier>: Term describing a subtype or view of the <object>

<object>: Term for self-contained real world (business) concept

If the <qualifier> is a unique term of its own and already contains the meaning of <object> the <object> is omitted.

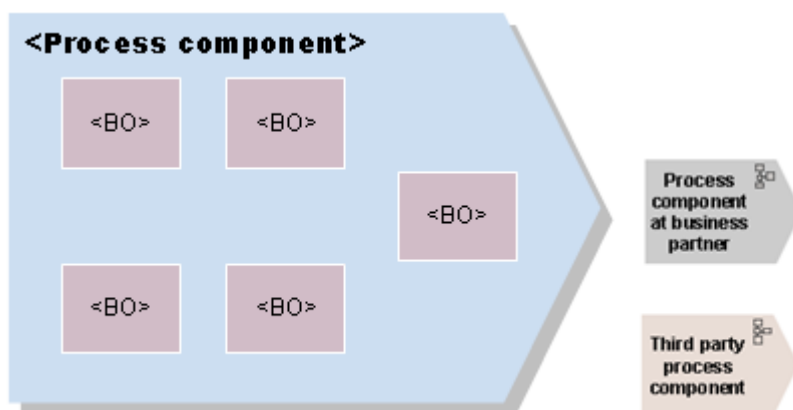
Examples:

- Payment Register (qualifier: Payment, Object term: Register)
- Customer ~~Business Partner~~ (qualifier Customer is unique term of its own, therefore term Business Partner is omitted)

4.3.2 Process Components

Process components describe self-contained parts of a value chain and group business objects together. One business object belongs to exactly one process component. Process components contain data and services for accessing that data and thus form reusable modules of larger applications. Access to data is modeled by service interfaces and operations. For example, the ERP process component **Sales Order Processing** provides the service interface **Manage Sales Order In**, among others. From this interface, you can access the service operation **Create Sales Order**.

Process Component combines related business objects and their A2A, B2B and A2X Service Operations and Service Interfaces.



Example:

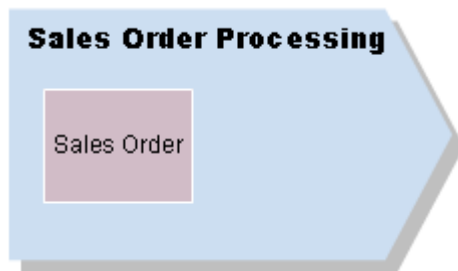


Figure 3: Process Component

Characteristics:

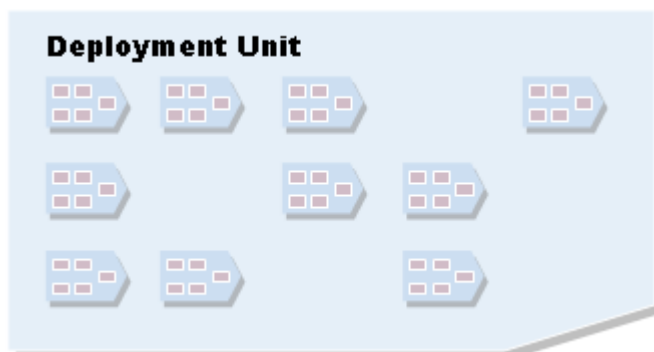
- A Process Component is a structuring and modeling construct. It is not a tangible entity in the development environment.
- A Process Component belongs to exactly one Deployment Unit.
- A Process Component contains one or more Business Objects.
- Larger pieces of a system's functionality (i.e. Scenarios) are assembled by using or reusing Process Components.
- There are two types of accesses for accessing the process component data:
 - Asynchronous: Asynchronous accesses are the means to choose either an A2A or B2B communication
 - Synchronous: This depends mainly on other components of the same application.

More Information:

SAP Help Portal: [Process Components](#)

4.3.3 Deployment Units

Deployment Units group process components that interact with each other and that are to be installed together on a system. Deployment units show the type of interactions needed between the process components within the deployment unit.



Example:

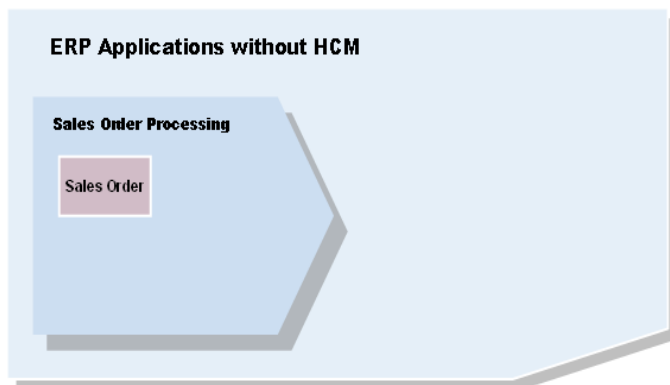


Figure 4: Deployment Unit

Best Practice Guideline:

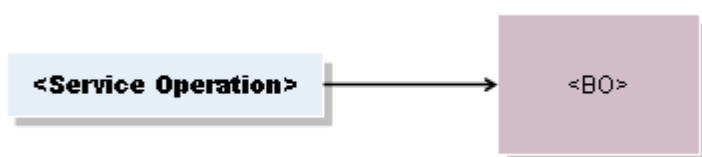
- A Deployment Unit is not an entity of physical software shipment and that means it does not necessarily form a piece of software that is shipped independently from other pieces.
- A Deployment Unit contains one or more Process Components.
- Process Components of a Deployment Unit are completely decoupled from other Deployment Units via enterprise services (A2A/B2B).
- A Deployment Unit can be replaced by other software components.
- A Deployment Unit can run in multiple instances connected to multiple instances of other Deployment Units.
- In practice this means mainly that you can run multiple local instances of one Deployment Unit connected to one central instance of another Deployment Unit.

More Information

SAP Help Portal: [Deployment Units](#)

4.3.4 Service Operations

Service Operations are entities that perform specific tasks on a business object, for example, creating, updating, or deleting a business object. The operation is a specification of a function with a set of message types assigned as the signature. An operation is assigned to exactly one business object, whereas a business object can have multiple operations. Depending on the type of access required to the data or business object, the operations can be asynchronous (for A2A or B2B communication) or synchronous (for access from other components of the same application). An example of an operation is “find sales order by order ID.”



Example:



Figure 5: Service Operation

Best Practice Guideline:

- A Service Operation name is unique within an interface.
- Operation naming rules apply for all categories of service operations (A2X, A2A, and B2B).
- Naming rule for Outbound Service Operation:
- Naming rule for Inbound Service Operation:

Syntax: <Transaction Activity> <Service View> <Action>?

Syntax: <Action> <Service View>

<Transaction Activity>: Activity of Operation such as Request, Confirm of, Notify of, Inform of, Query, Respond

<Service View>: Service View that defines the operation signature. Omitted if the name of the service interface covers the semantics of the service view

Syntax: <qualifier><object>?<component>?*

<qualifier>: Term expressing a semantic restriction of the relevant instances for the service

<object>: Name of business object the service view is derived from

<component>: Name of sub structure of the object

If the *<qualifier>* is an established term and covers the semantics of *<object>*, then *<object>* is omitted.

Example of Service View: Purchase Order Delivery Terms

<Action>: Business Function e.g. Create, Update, Cancel, Maintain.

Examples:

Outbound:

Notify of Invoice (with Transaction Activity "Notify of" and Service View "Invoice")

Inform of Purchase Order Cancel (with Transaction activity "Inform of", Service View "Purchase Order" and Action "Cancel")

Inbound:

Maintain Invoice (with Action "Maintain" and Service View "Invoice")

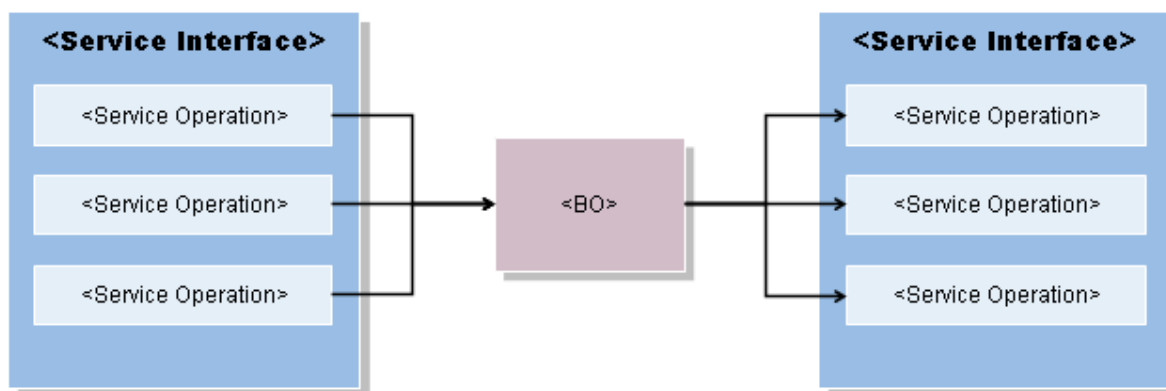
Create Invoice based on Attachment (with Action "Create", Service View "Invoice" and Object "Attachment")

More Information

SAP Help Portal: [Service Operations](#)

4.3.5 Service Interfaces

Service Interfaces are named groups of operations. A service interface belongs to exactly one process component, whereas a process component can contain multiple service interfaces. Service interfaces specify offered (inbound service interfaces) or used (outbound service interfaces) functionality. When you model the operations and service interfaces, you use predefined patterns to ensure that the naming and definition of your services are unified. These interface patterns are derived from the access type needed and cover the majority of use cases. Thus, service interfaces, operations, and message types are always modeled in the same way.



Example:

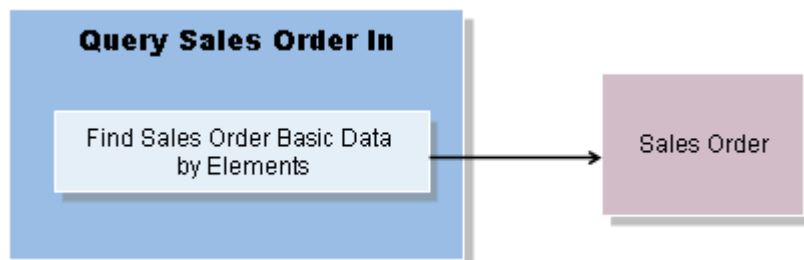


Figure 6: Service Interface

Best Practice Guideline:

- A Service Interface name is unique within a Process Component.
- Naming Rule:
 Syntax: <Interaction activity><Direction>
 <Interaction activity>: Verb or verb phrase in the “-ing” or in substantive form
 <Direction>: Fixed term “In” or “Out”
 Examples: Query Sales Order In, Request Invoicing Out, Fulfillment In

More Information:

SAP Help Portal: [Service Interfaces](#).

4.4 Interface Patterns

Interface patterns are defined to ensure the behavioral integrity of the Enterprise services. When you model the Service Operations and Service Interfaces, use predefined patterns to ensure that the naming and definition of your services are unified. Interface patterns define naming rules for operations and service interfaces based on the business object and its node structure, grouping rules for the operations in the service interfaces, naming rules for the message types, and rules for the message choreography of an enterprise service interaction.

The following Interface Patterns are available:

4.4.1 Modeling A2A Interactions

Interface patterns for A2A interactions model the message exchange between two process components with the help of asynchronous and synchronous operations. These patterns can also be used for B2B communication. For example, a “request-confirmation” pattern describes semantics and naming conventions for an interaction in which a change or update is requested and a confirmation is expected to be returned.

You need to decide which interface pattern best suits your use case and choose from the following interface patterns:

Asynchronous Interface Patterns:

Request Confirmation (Bi-directional):

This interface pattern should be used for bi-directional, asynchronous A2A interactions. This pattern is the most common pattern.

The typical use case for this pattern is the request and update of an order.

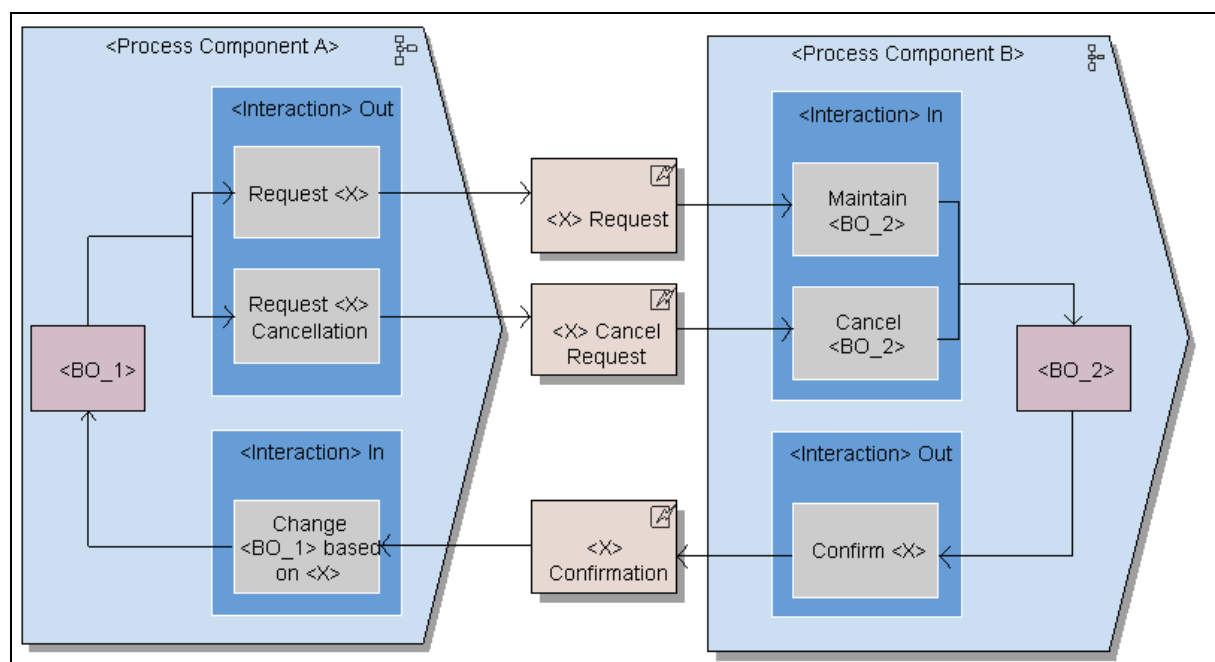


Figure 7: Request Confirmation

One process component is requesting something from another process component and gets a confirmation once the requested business action is performed. The confirmation is not just a technical confirmation that the message has been received.

Notification (One-directional):

This interface pattern should be used for one-directional, asynchronous interactions. For this scenario the main focus is a goal-oriented notification which the customer is waiting for. Notification means that process component A has the obligation to notify Process Component B, but is not interested in any response. The notification message is specific to the receiver.

Naming of the receiving Operation depends on what is happening on the receiving side: Create <BO>; Change <BO> etc.

A typical use case would be the notifications about the progress of an internet purchase.

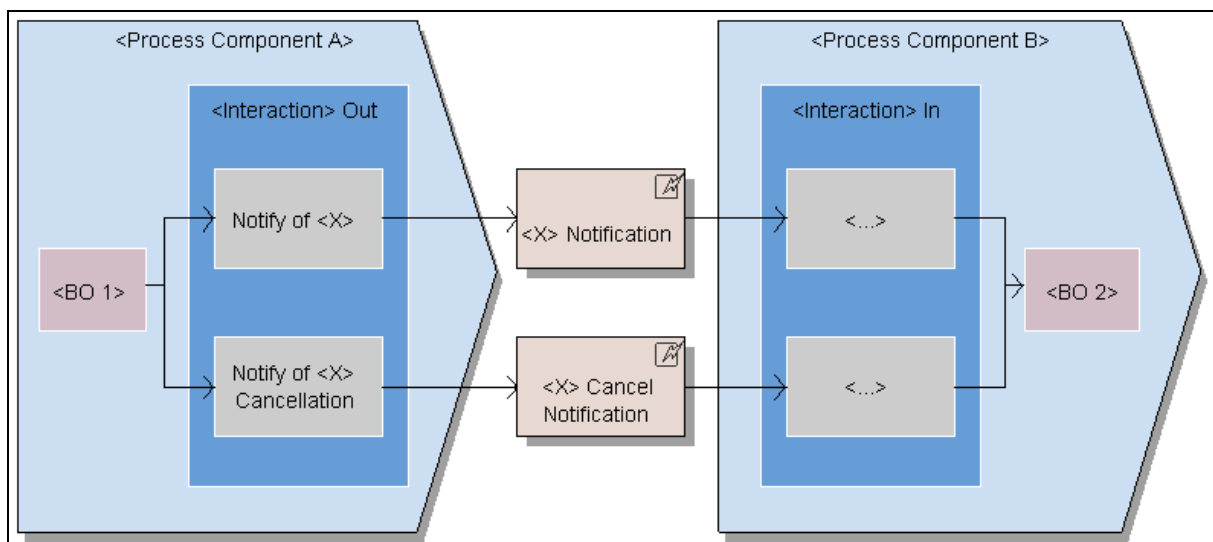


Figure 8: Notification

Information (One-directional):

This pattern should be used for A2A pub/sub like scenarios.

The sender does not know at all what the receiver is doing with the message. This means the sender does not know, if the receiver is actually interested in the event of the given BO_1 instance. Therefore there is a high risk of sending a lot of messages that are actually ignored at the receiver. This pattern should only be used if there is no other choice.

Since the sender does not know the receiver, this interface has to be defined like an A2X interface. You simply group all events of one Business Object in one Interface. There is no specific interaction like in other A2A interfaces since you do not know the receiver.

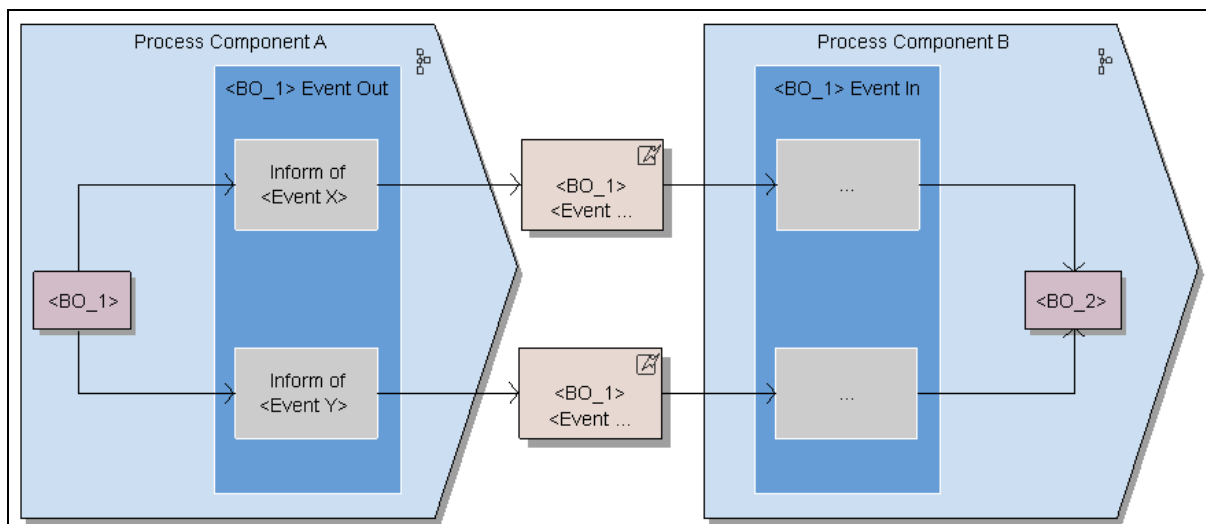


Figure 9: Information

Best Practice Guideline:

Criteria for using this pattern are:

- There are already multiple receivers for the same event (which do actually different things with the same information).
- The sending system is often non-SAP software, which is not capable of evaluating whether the receiver actually is interested in the message or not.
- Evaluation whether the message is relevant would only be possible, if (master) data is replicated to the sender, which the sender does not need otherwise.
- The sender does not care about the receiver and has also no obligation to inform the receiver (notification pattern).
- Do not use this interface pattern to distribute master data.

Synchronous Interface Patterns:

- Query Business Object in A2A Communication:
This interface pattern should be used for synchronous communication to read data from a business object by means of a selection criteria and not using a known ID.
- Read Business Object in A2A Communication:
This interface pattern should be used for synchronous communication to read data from a business object for which an ID is known.

More Information:

- SAP Help Portal: [Synchronous Interface Patterns for A2A Interactions](#)

Best Practice Guideline:

- Since these two interface pattern works with synchronous communication, they must not be used to update transaction data or to distribute master data.
- They should be restricted to read-only access.

4.4.2 Modeling A2X Services

A2X services are used to model synchronous access to data from UI components or from other clients.

You need to decide which interface pattern best suits your use case and choose from the following interface patterns. These interface patterns ensure harmonized service interfaces across all the business objects.

Manage Business Process Object:

This interface pattern should be used for modeling application accesses that do not need to be persisted. It describes the synchronous access to the BO from a UI component or from other clients. It defines rules for naming Enterprise Service Operations and Service Interfaces, rules for grouping enterprise service operations into service interfaces, rules for naming message interfaces, and rules for the message choreography of an Enterprise Service interaction.

The Figure shows a “Manage BO Pattern” that is to be used for the interaction with a particular Business Object. It gives the information on how to define the interface, operations, and message types used.

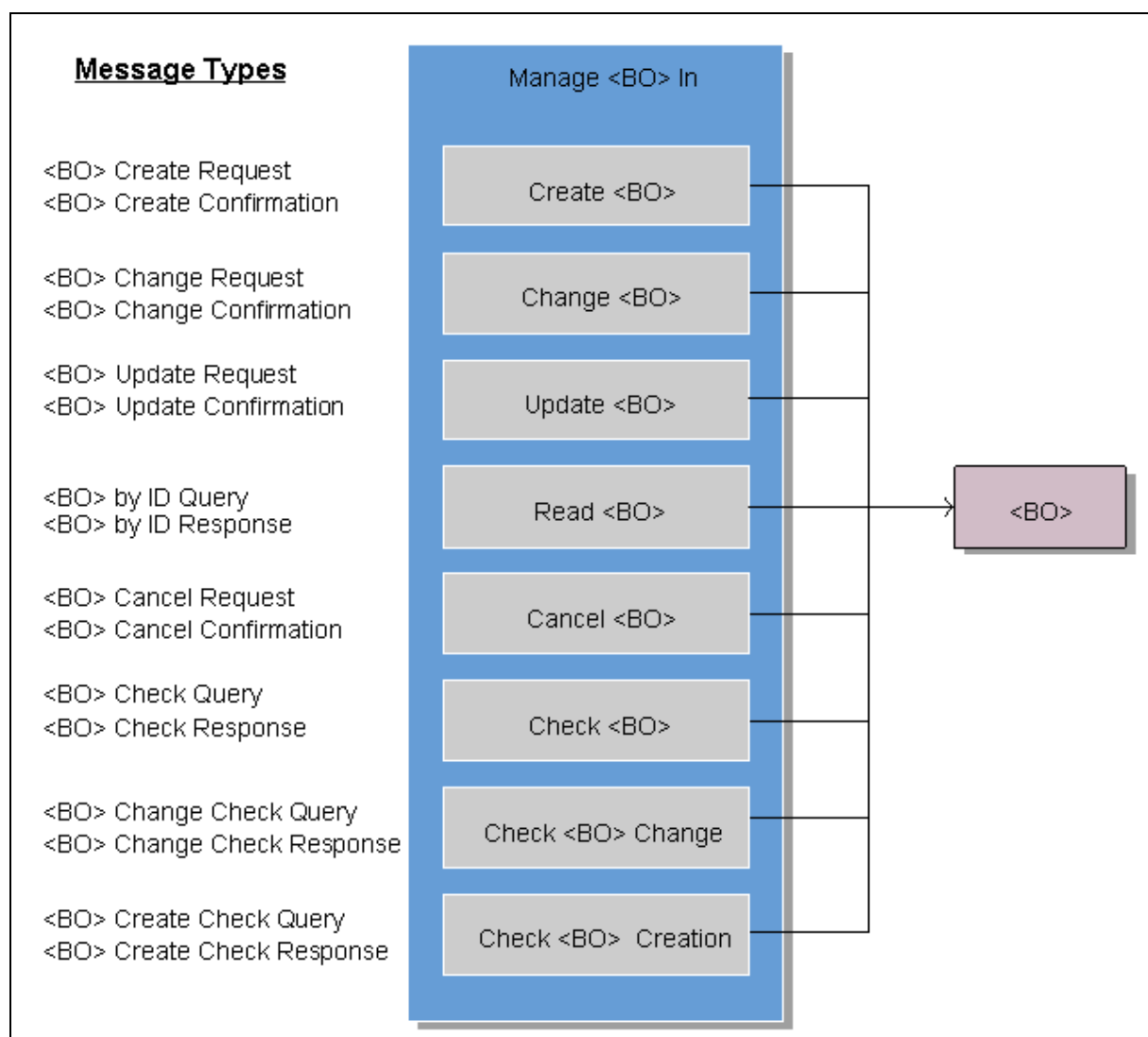


Figure 10: Manage Business Process Object

Best Practice Guideline:

Read <BO>

- The read operation should be implemented in a way that it can handle a list of IDs as input.
- There should be no separate operations for reading different parts of the object except in the case where object mixes the master data parts (in header) and transactional data parts.
- Naming example:
 - Operation: **Read Purchase Order**
 - Message Type Request: **Purchase Order by ID Query**
 - Message Type Response: **Purchase Order by ID Response**

Create <BO>

- Naming example:
 - Operation: **Create Purchase Order**
 - Message Type Request: **Purchase Order Create Request**
 - Message Type Confirmation: **Purchase Order Create Confirmation**

Change <BO>

- Change operation follows the “Last one Wins” strategy. It applies changes to the data ignoring if changes were applied to the same data since read by the calling consumer.
- Use this very carefully and it is recommended to favor Update operation instead of change.
- Naming example:
 - Operation: **Change Purchase Request**
 - Message Type Request: **Purchase Request Change Request**
 - Message Type Confirmation: **Purchase Request Change Confirmation**

Update <BO>

- Update operation follows “First One Wins” Strategy and changes data only if no changes were occurred since the data was last read. It checks for concurrent updates and sends an error message if someone else has changed the message.
- You should implement update operations in cases with high probability of concurrent updates, or in cases where the “last one wins” of change operation is not acceptable.
- Naming example:
 - Operation: **Update Sales Order**
 - Message Type Request: **SalesOrderERPUpdateRequest_sync**
 - Message Type Confirmation: **SalesOrderERPUpdateConfirmation_sync**

Check <BO>

- Check operations should be set up independently from the change, create or cancel operation.
- The check operation should also include the check if the object is locked or not.
- In case you need different check operations for create and change, it is allowed to create different ones.
 - Operation: **Check <BO> Creation** or **Check <BO> Change**

- Message Type: **<BO> Check Creation Query** or **<BO> Check Change Query**
- Naming Examples:
- Operation: **Check Sales Order Creation**
- Message Type Query: **SalesOrderERPCreateCheckQuery_sync**
- Message Type Response: **SalesOrderERPCreateCheckResponse_sync**

Manage Master Data Object:

This interface pattern is used for master data object type business objects (BOs). To change or read data, you do not normally access the master data objects with a large operation.

More Information:

- SAP Help Portal: [Manage Master Data Object](#)

Best Practice Guideline:

SAP recommends that you group the master data into meaningful categories and model an operation for each group.

Query Business Object:

This interface pattern describes the search for data records by means of selection criteria that are handed over to the operation. It is read-only access. Divide the operation up according to the purpose of the search and then by which selection criteria those are to be used.

More Information:

- SAP Help Portal: [Query Business Object](#)

Business Object Action:

This interface pattern is used to perform actions that operate on business object (BO) data. There are typically only a few parameters contained in the the request so that the action can be performed.

More Information:

- SAP Help Portal: [Business Object Action](#)

4.5 Transaction Communication Patterns

Transaction Communication Party (TCP) describes an atomic dialog between a Sender and Receiver. Typical TCPs used by Enterprise Services are:

- Query/Response: means that messages are sent back and forth but that the state maintained by the service does not change.
- Request/Confirmation: means that messages are sent back and forth and that the state maintained by the service may change.
- Notification: means that a service sends a message that contains a notification of an event.
- Information: means that a service sends a message containing other information.

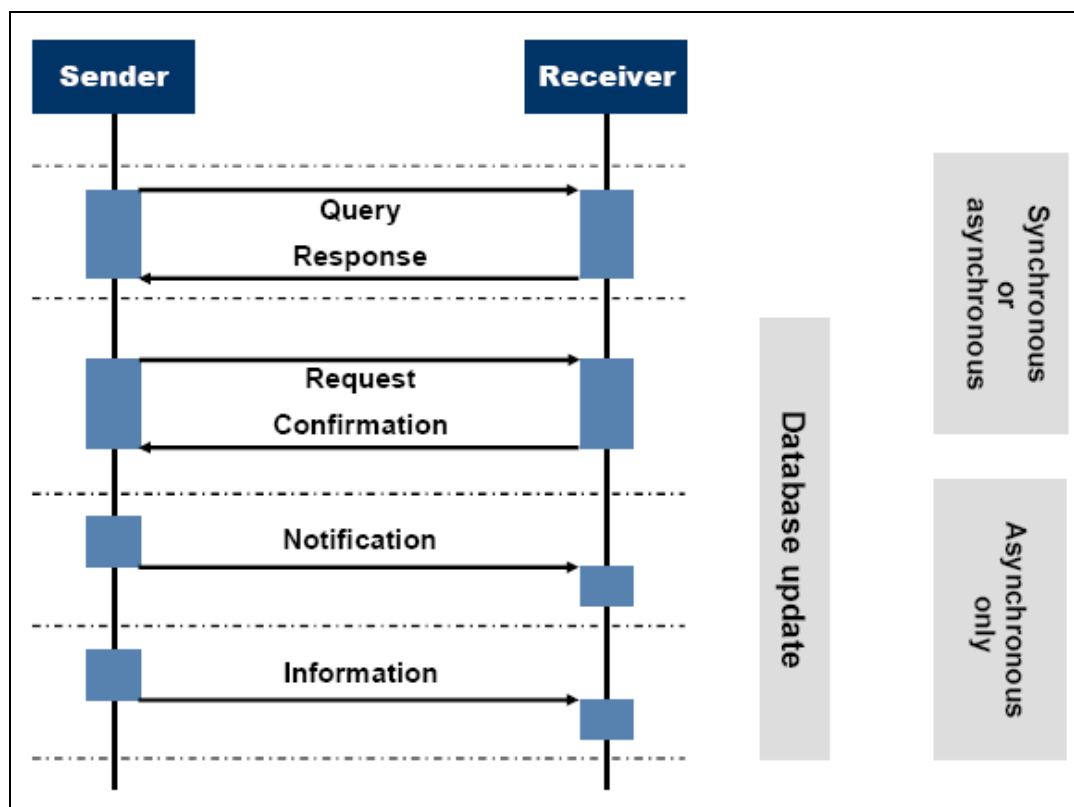


Figure 11: Transaction communication Pattern

Best Practice Guideline:

The selection of a TCP should take place based on the classification of patterns. The communication pattern to be used should be derived from the business perspective, depending on the expectation of the sender of a message and the obligation of the recipient to react to it. The following diagram illustrates the decision tree for this:

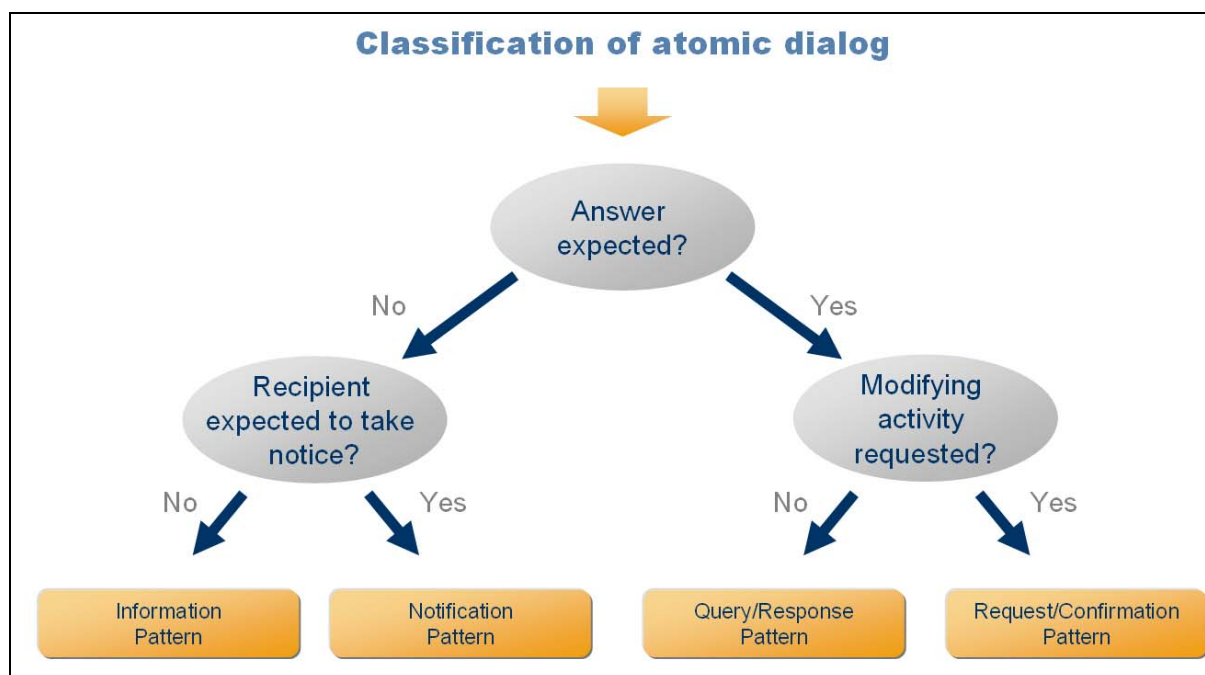


Figure 12: Decision Tree

5. Modeling Services in ESR

Modeling focuses on design and ensures reusability, naming conventions, and scalable interactions and integration scenarios. The Enterprise Services Builder in the ESR is used for defining and managing the objects necessary for implementing an Enterprise Service. You first need to create models in order to determine the essential business objects such as data types, message types, and service interfaces necessary for your application in the ESR. These objects are then used to generate and implement your services.

Best Practice Guideline:

It is recommended to follow the below when modeling the services in ESR:

- Import a Software Component Version (SWCV) from the SLD into the ESR and define a namespace for it. All objects required will be created under the namespace.
- Create a new Process Component Model (SAP ProComp Model).
- Add the required business objects to the model.
- Identify and model the operations and service interfaces that will provide the process components with access to the data. When modeling the services, use the predefined patterns.
- To access the data of other process components, use Process Component Interaction model (SAP Procomp Interaction model). Identify and model the required operations and service interfaces.
- Use the Interaction Scenario model to show interactions with other model.
- Active all your changes and release the service interfaces.

5.1 Model Types in ESR

According to the harmonized Enterprise Service model, there are five model types namely Process Component model, Integration Scenario model, Process Component Interaction model, Business Object map, and integration scenario catalogue.

5.1.1 Process Component Models

The Process Component models in ESR enable SOA governance and help to understand the business semantics of enterprise services in the business process platform. They provide the details of the internal structure of a process component. It is a view on the Service Interfaces and Service Operations provided by a Business Object. You use one or multiple Business Objects to model the data. Service operations (A2A, B2B, and A2X) and service interfaces are defined in the Process Component models. For this reason, they are also called the Provider Views. Process Components models show all possible interactions with other Process Components.

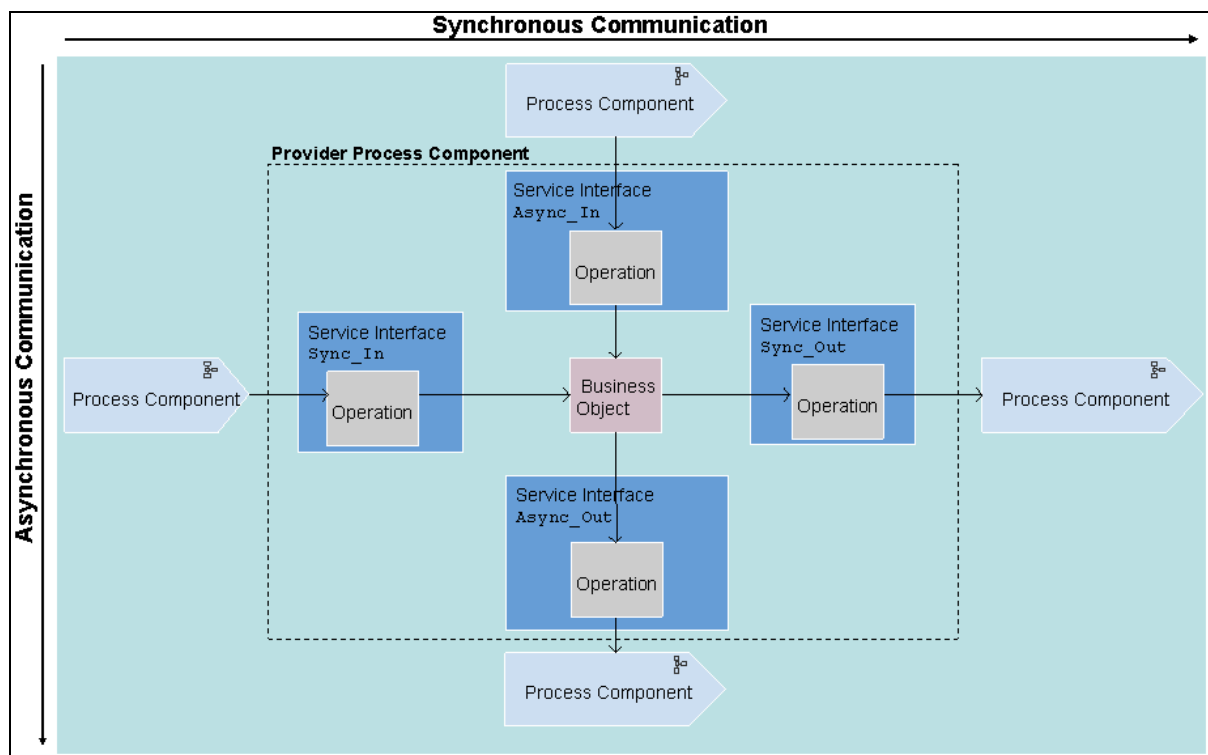


Figure 13: Process Component model

Best Practice Guideline:

- In the process component model, the process component should be modeled from the provider view.
- To keep the model transparent, model synchronous operations from left to right and asynchronous operations from top to bottom.
- SAP recommends that you use the interface patterns for synchronous access to BO data. You need to select from the following interface patterns for modeling the data:
 - Manage Business Process Object
 - Manage Master Data Object
 - Query Business Data Object
 - Business Object Action



Note

These interface patterns are discussed above in Section 4.4.2.

5.1.2 Integration Scenario Models

Integration Scenario models describe which process components belongs to which deployment units and how the process components interact with each other in an end-to-end scenario. The integration scenario models give a better understanding of the whole process. Communication between Process Components is reflected and characterized through connection types.

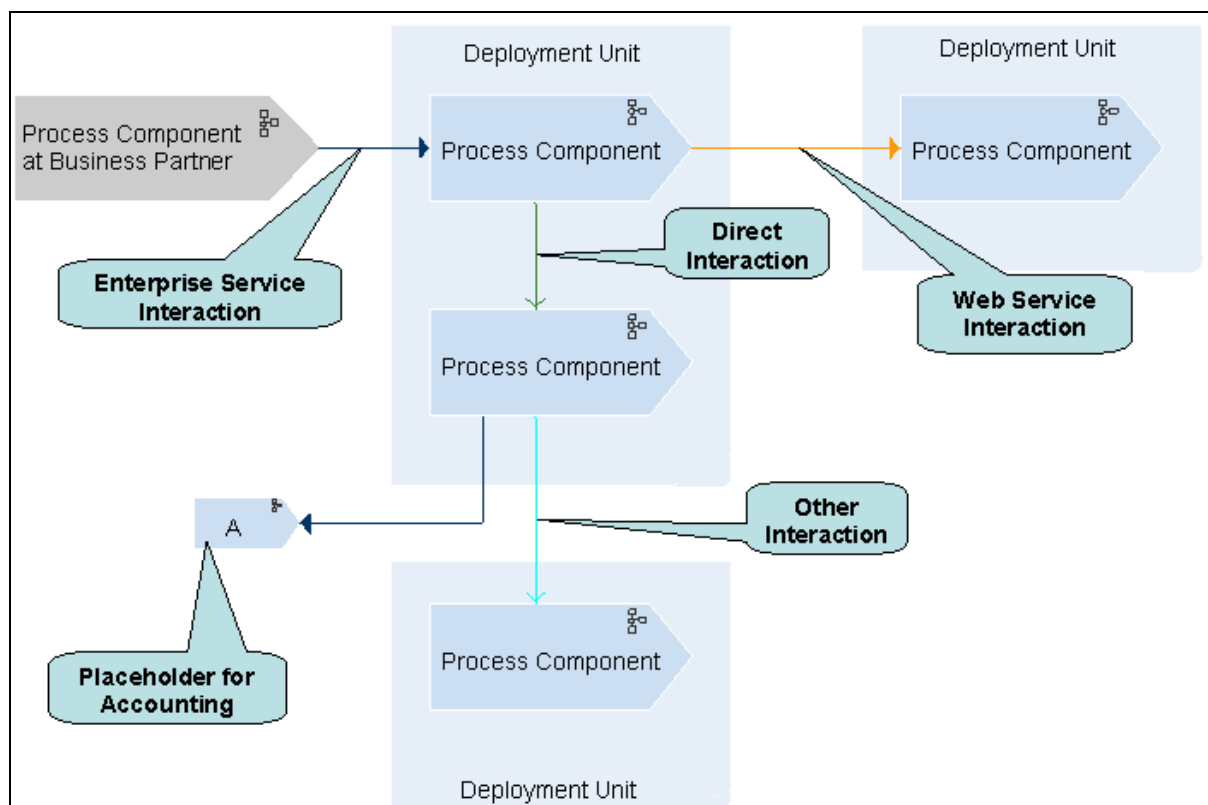


Figure 14: Integration Scenario Model

Best Practice Guideline:

The following Process Components types can be used in this model:

- Process Component (PC that you implement)
- Process Component at Business Partner (PC at a Business Partner is connected to PC in your business via B2B communication)
- Third-party Process Component (A third party PC is connected to PC in your business via A2A communication)

You can use the following four types of interactions between the process components that show how two process components exchange between each other:

- Enterprise service interaction (implemented outbound from service interfaces in the ESR)
- Web service interaction (represents synchronous point-point interaction)
- Direct interaction (interaction is done via method calls or function module calls. Example: RFC call)
- Other interaction (other interactions between deployment units, different from web service or enterprise service interactions)

5.1.3 Process Component Interaction Models

Process Component Interaction Models represent an interaction with a specific business goal between two Process Components. The model shows all the involved Business Objects, service interfaces, operations, message types. Process Component Interaction Model can only be used for an enterprise

service interaction and there can be for none, one, or multiple process component interaction models assigned to one enterprise service interaction.

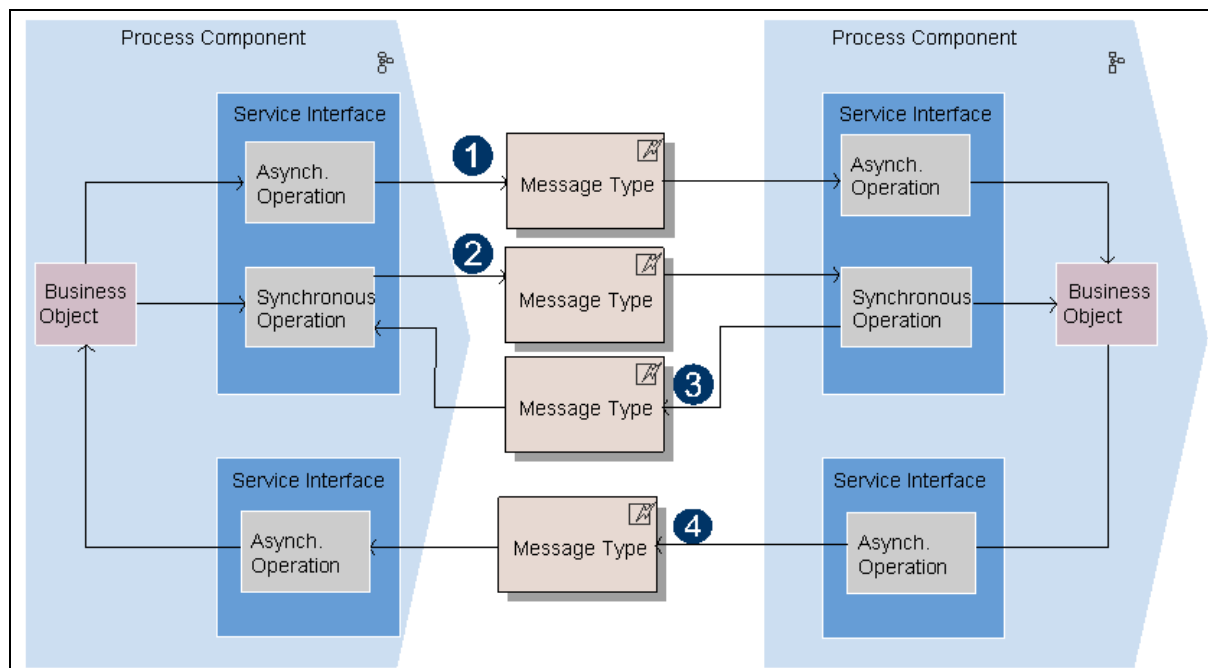


Figure 15: Process Component Interaction Model

Best Practice Guideline:

In designing this model, you should use the following message types for communication:

- One message type for an asynchronous operation without mapping.
- Two message types for a synchronous operation without mapping (one for the request and one for the response).
- If the message type structures from the sending and receiving process components do not agree, the message types must be mapped on to each other using a mapping. You therefore need an outbound and a target message type, rather than just a message type.

SAP recommends that you represent the sequence of events from **top to bottom**.

The following interface patterns are recommended to be used in this model:

- Asynchronous Interface Patterns:
 - Request/confirmation
 - Notification
 - Information
- Synchronous Interface Patterns:
 - Query Business Object in A2A communication
 - Read Business Object in A2A communication

Note

These interface patterns are discussed above in Section 4.4.1.

5.1.4 Business Object Map

Business object maps (SAP entity map), or business object template maps, aggregate business objects or business object templates in a model for overview purposes. A business object map is an entity map, which is a structured directory of all entities of the main entity types. An entity map for a given application is a structured directory of all deployment units, process components, and business objects in the application. Business object maps are defined for all major applications and are shipped as ESR content.

The Business Object map is shipped as ESR Content.

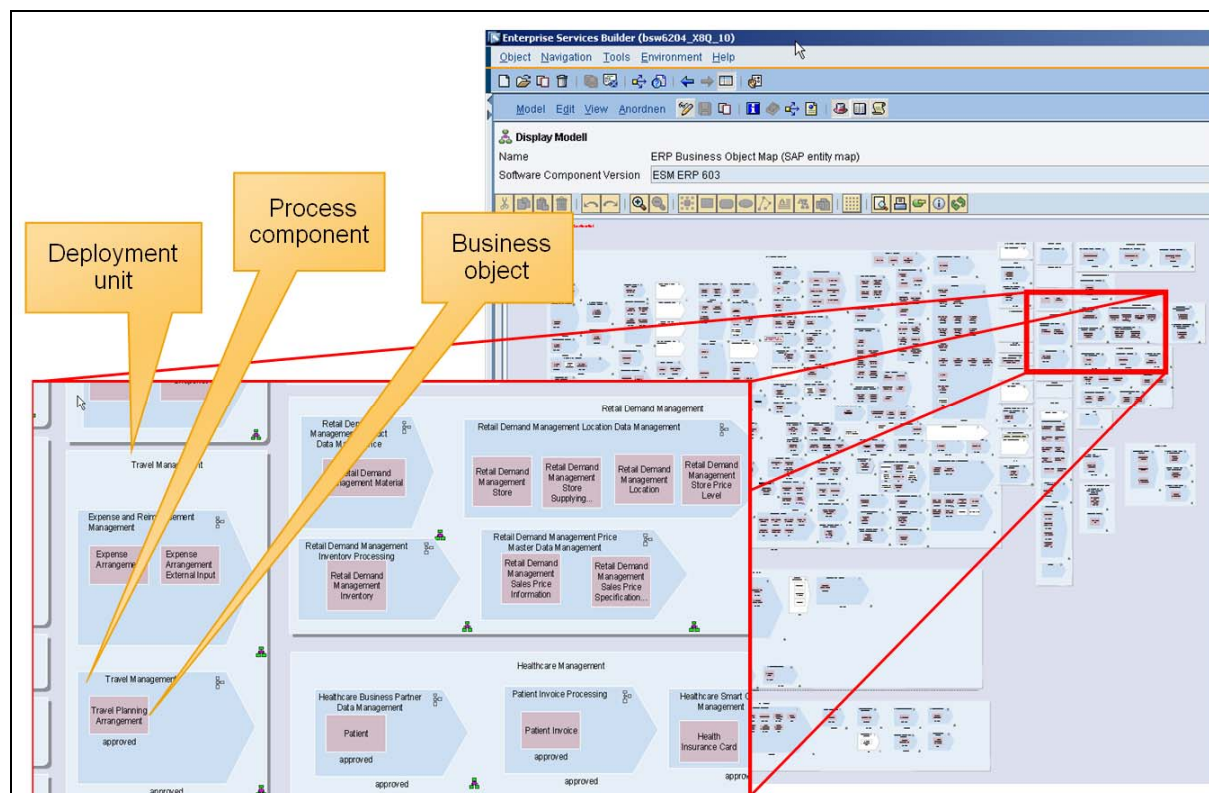


Figure 16: Business Object Map

5.1.5 Integration Scenario Catalogue

Integration scenario catalogs (SAP Scenario Catalogue) group and structure all the integration scenarios of a solution and thereby represent the business starting point for process modeling. The contained scenarios and their variants are clustered using integration scenario groups. You can navigate from a catalog to all contained elements.

6. Process Integration Scenarios

The process integration scenarios provide an overview of the process flow for a collaborative scenario. These scenarios bind together all relevant objects defined in the Enterprise Services Repository and can be used in the Integration Directory for configuring an A2A or B2B scenario. The process integration scenario is the umbrella that brings together the objects required for the execution of a specific scenario and that identifies their interdependencies by modeling the process flow.

The Process Integration Scenario consists of:

- Application Components
- Actions
- Operations
- Connections

6.1 Application Components

Each process integration scenario contains two or more application components. An application component typically represents a business partner or a component within the business partner landscape.

6.2 Actions

An action is an object in SAP NetWeaver PI that represents a function within an application component that is not subdivided further. Actions divide up the process flow of an integration scenario. Actions in different application components can exchange messages with each other within an integration scenario.

6.3 Connections

A connection is a link between two actions within a Process Integration scenario.

Connections are of two types:

- Sequence
Sequence represents the sequence of two actions within the same application component. A sequence is required for portraying the Process Integration Scenario process, however it contains no additional information.
- Cross-Component Connection
A cross-component connection connects actions from different application components with each other. These actions exchange messages with each other in a Process Integration scenario. A cross-component connection defines the interfaces and mappings used.
Cross-component connections differentiate between synchronous and asynchronous communication

Best Practice Guideline:

- A Process Integration scenario models the complete exchange of messages for a collaborative process and provides an overview of the process flow. The advantages of defining Process Integration Scenarios in the Enterprise Services Builder (design tool) are as follows:
 - The Process Integration Scenario provides you with an overview of the process and the process flow.
 - The Process Integration scenario combines all objects that are involved in this process: interface objects, mapping objects, executable integration processes from the Enterprise Services Repository, and product versions from the System Landscape Directory.
You can access all of these objects from the Process Integration Scenario.
 - The Process Integration Scenario contains all design time information about the process that is required for its configuration. If you have defined a Process Integration scenario, you can use this as a template for configuration. The relevant objects are predominantly created automatically. For more complex collaborative processes the use of Process Integration scenarios enables you to clearly design the configuration.

www.sdn.sap.com/irj/sdn/howtoguides