Copyright

© Copyright 2008 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.
# Table of Contents

1. Introduction
   1.1. Why Enhance an Enterprise Service? ................................................................. 1
   1.2. How Do You Enhance an Enterprise Service? .................................................... 1
   1.3. Where to Enhance an Enterprise Service within the Message Structure .......... 2

2. Developing in SAP NetWeaver PI
   2.1. Developing in SLD ............................................................................................ 6
       2.1.1. Creating a Product, Product Version, Software Unit, SWC and EnSWCV in SLD .... 6
       2.1.1. Creating SWC and EnSWCV in SLD ......................................................... 8
   2.2. Defining Dependencies Between SWCV and EnSWCV in SLD ......................... 9

3. Developing in ESR
   3.1. Importing an EnSWCV into ESR ..................................................................... 13
   3.2. Creating a Namespace in EnSWCV ................................................................. 15
   3.3. Creating an Enhancement Data Type in the SWCV .......................................... 16

4. Developing in the Backend System
   4.1. Generating an Enhancement Proxy Structure ................................................... 20
   4.2. BAdI Implementation ...................................................................................... 24
   4.3. Enhancing Synchronous Enterprise Services ................................................. 24
       4.3.1. Scenario 1-Enhancing Service Signature and Mapping to Existing Implementation ... 25
       4.3.2. Scenario 2-Service Enhancement Requiring Application Enhancement .......... 27
   4.4. Enhancing Asynchronous Enterprise Services ................................................ 30
       4.4.1. Example of Asynchronous Enterprise Services .................................... 30

5. Appendix
   5.1. Relationship between Products, Product Version, SWCs, and SWCVs in SLD .... 32
   5.2. Creating an Enhancement Structure in EnSWCV ............................................. 33
       5.2.1. Using a Data Type from an Underlying SWCV ........................................ 34
       5.2.2. Using a Data Type from the Enhancement Namespace ............................ 35
       5.2.3. Using a Built-In Data Type Integrated in ESR ........................................... 36
   5.3. Impact of Update or Upgrade of an Underlying SWCV on an Enhancement ....... 37
   5.4. Documentation ............................................................................................... 37
   5.5. Glossary ......................................................................................................... 38
1. Introduction

Enterprise Services allow you to leverage SAP solutions to include a wide range of composite applications that are provided by customers and partners to build new, flexible, and innovative solutions. Enterprise Services are highly-integrated Web services combined with business logic and semantics that can be accessed and used repeatedly to support a particular business process. SAP publishes all Enterprise Services that support SAP Business Suite functionality on the Enterprise Services Workplace in the SAP Developer Network (SDN). For more information, see the Enterprise Services for SAP, ERP 6.0 as listed in Documentation.

This guide explains how to enhance SAP Enterprise Services to accommodate your business requirements.

Note: The screenshots in this guide are from the SAP NetWeaver PI 7.0 platform and may differ from the SAP NetWeaver PI 7.1 screens.

You define a service operation enhancement in the Enterprise Service Repository (ESR). Based on this, you generate an enhancement proxy and implement the required business logic at your back-end system. You implement the business logic in an Enterprise Service-specific Business Add-In (BAdI). Advanced knowledge of SAP XI 3.0 is necessary to understand this document. Note that you can use the ESR which resides within SAP NetWeaver Process Integration (SAP NetWeaver PI 7.0) or within SAP NetWeaver Composite Environment (SAP NetWeaver CE 7.1). This document is written using an ESR of SAP NetWeaver PI 7.0. For more information about SAP Process Integration, see SAP Service Marketplace at https://service.sap.com/xi.

1.1. Why Enhance an Enterprise Service?

There are a number of reasons why you may need to enhance an Enterprise Service:

- The business application has been extended (by SAP) and these extensions are also offered in enterprise services
- Customers or partners have enhanced SAP’s business functionality and want to offer additional elements in existing services
- Standard service does not expose all fields of a business object and additional fields are needed in the service signature.

1.2. How Do You Enhance an Enterprise Service?

There are three different ways to enhance an Enterprise Service; however only the second and third are relevant for partners and customers. This guide focuses on the second alternative as it is the one that SAP recommends to you.

1. Extend the original standard service with new (optional) fields; this alternative is used within SAP, typically when an industry-extension is added.
2. Enhance standard service in a new enhancement name space; this alternative is used by customers and partners. They define their enhancement elements in their own namespaces and these enhancement elements refer to the SAP standard Enterprise Service. This is illustrated in alternative 2 in the figure below.
3. Duplicate service as a new and specific, and add the required fields. The drawback of this alternative is that the reference link between the customer enhancements and the SAP standard Enterprise Service does not exist anymore. Thus, any enhancements or corrections provided by SAP in a future release will not be available for the customer’s enhanced service.
The figure below illustrates the three alternatives to enhance an Enterprise Service:

**Figure 1: How to Enhance an Enterprise Service**

1.3. Where to Enhance an Enterprise Service within the Message Structure

If you use enhancements in your back-end system that you wish to transfer using an Enterprise Service, this chapter explains how to choose the right part of the structure of the Enterprise Service to enhance.

A message typically contains a message header, a business document and an attachment. The business document contains one or more nodes that are themselves containing elements.

The figure below illustrates the general structure of a message:

**Figure 2: General Structure of a Message**
You can enhance either the message data type or one of the nodes of the business document. We recommend you add your enhancement element to the node to which it semantically belongs. For example, if you want to add a buyer party to a service operation, you attach it to the existing node party. If there is no node that is semantically close to your enhancement element, you can also create your own node.

The figure below illustrates how Enterprise Services are enhanced, either by enhancing the whole message data type (1) or by enhancing a certain node (2):

![Diagram showing Enterprise Service Enhancement](image)

**Figure 3: Model of Enterprise Service Enhancement**

**Note:** The node you are enhancing must be typed with an explicit ESR data type. We recommend that this data type is used only once in this service, otherwise your enhancement element will appear in all the services using this data type. You can check this using the Where-Used pushbutton in ESR.

Communication between the service provider and service consumer takes place based on the message interface that is defined by input and output message types. The message types are defined by message data types. Begin by selecting the message data type that has to be enhanced. For example, you might want to enhance the data type *output message*. Next, check the structure of this message data type to determine the suitable node for the enhancement.

Enterprise Services message data types usually have the same overall structure, as shown in Figure 2.

**Note:** To avoid problems, we recommend that you enhance only those data types that are used within a single Enterprise Service. If you have to enhance a data type that is used in multiple Enterprise Services, you should implement the corresponding BAdIs in all the affected Enterprise Services to ensure consistent behavior.

The figure below illustrates where to find the function "where-used" to check in how many enterprise services a chosen data type is used:
Example

The service operation Read Customer Contract by ID uses the customer contract number in the back-end to read customer contract information. You send a query containing the ID of the customer contract to the back-end system. In the back-end, the detailed information about the customer contract is read and posted to the consumer of the service. You want to enhance the response with information about the user who last changed the customer contract.

The structure of the response is represented with the message type CustomerContractERPByIDResponse_sync. The subnode SystemAdministrativeData exists within the root node CustomerContract. This subnode contains the elements CreationDate, CreationUserAccountID, and LastChangeDate. This node is semantically suitable for enhancing the service operation with the element that indicates the last changes that were performed to the customer contract. You can name this element (for example, LastChangeUser). Within ESR, find the data type that is assigned to the node SystemAdministrativeData and enhance it using the enhancement concept. The enhancement consists only of the element LastChangeUser.
The procedure for creating the enhancement and implementing the BAdI in an application system is depicted in the following figure:

Figure 4: Creating Enhancements and Implementing Business Add-Ins (BAdIs)

For more information about BAdIs, see Business Add-Ins (BAdIs) as listed in Documentation.
2. Developing in SAP NetWeaver PI

Development in the SAP NetWeaver PI has two phases:

1. Developing enhancements in the ESR
2. Creating enhancement proxies and implementing BAdIs in the back-end system

2.1. Developing in SLD

The software component version (SWCV) that you create to enhance Enterprise Services is called an enhancement software component version (EnSWCV). You can create an EnSWCV in the system landscape directory (SLD) in SAP NetWeaver PI by following the steps below.

2.1.1. Creating a Product, Product Version, Software Unit, SWC and EnSWCV in SLD

**Step 1: Creating a Product and a Product Version**
Create a non-SAP product with a name that describes the enhancement. For examples and information about the relationship between product, product versions, software component (SWC), and SWCV see section 5.1. Follow the steps below to create a new product and product version.

1. On the tools entry page (Exchange Infrastructure Tools 7.0/Composite Enviornment Tools 7.1), choose **System Landscape Directory**.

2. Under **Software Catalog** choose **Products**.

4. Enter Name, Vendor, and Version for the product.

   **Note:** Enter a URL such as www.customer.com in Vendor field. For more information, see *SAP Exchange Infrastructure: Guide for Customer Developments and Modifications* in the Integration Repository, Release SAP XI 3.0 listed in Documentation.

5. Choose Create.

The product and new product version are created in SLD.

**Step 2: Creating a Software Unit**

The following screen appears after you complete Step 1. Here, you enter the Name for your new software unit and choose Create.

The new software unit is created in SLD.
2.1.1. Creating SWC and EnSWCV in SLD

If a product exists, you can create a SWC and an EnSWCV during the maintenance of the product or independently during the maintenance of the SWC. The procedure for creation during the maintenance of products is described here.

**Step 3: Creating SWC and EnSWCV During the Maintenance of Products**

1. Create a non-SAP product and its version as shown in Step 1. Under Software Catalog choose *Software Components*.

2. Choose *New Software Component Version*.

3. Choose the non-SAP product version and software unit for which the EnSWCV must be created. For information on creating a non-SAP product version and software unit, see Step 1 and Step 2.
4. Enter the Vendor, Name, and Version of the product.

**Note:** To import the EnSWCV into the ESR immediately after its creation in the SLD, use the production state *Released*.

5. Choose Create.

The EnSWCV is created in SLD.

2.2. Defining Dependencies Between SWCV and EnSWCV in SLD

SAP NetWeaver PI allows you to create dependencies between one or more components. If you enhance an interface object, you must create a “based-on” dependency between the EnSWCV and the underlying SWCV, because one component can be a prerequisite for another component and some objects from the underlying SWCV can be reused in the original SWCV. For example, SWCV SAP SCM Basis 5.1 is the underlying component version for SAP SCM 5.1 because it (SAP SCM 5.1) uses some SAP SCM Basis classes, functions, and data types.

The process described above is depicted in the following figure:

![Dependency Between SAP SCM Basis 5.1 and SAP SCM 5.1](image)

**Figure 5:** Dependency Between SAP SCM Basis 5.1 and SAP SCM 5.1

If dependencies exist in ESR, you see an additional component, *Basis Objects*, in the navigation tree for the based-on SWCV. These objects contain all the namespaces from the underlying SWCV.

Once you create a dependency between the EnSWCV (as based-on SWCV) and the SWCV from which you need to enhance the service operation (underlying SWCV), you can reference objects from the based-on SWCV and also from underlying SWCV. This allows you to use the enhanced objects from namespaces of the underlying SWCV.
The dependency between the EnSWCV and underlying SWCV described above is illustrated in the following figure:

![Dependency Between Enhancement SAP SCM 5.1 and SAP SCM 5.1](image)

**Figure 6:** Dependency Between Enhancement SAP SCM 5.1 and SAP SCM 5.1

**Step 4: Defining Dependencies Between an EnSWCV and an Underlying SWCV**


2. Choose the EnSWCV that you created in Step 3.


5. In the list that appears, choose the SWCV for which you want to define the enhancement.

6. Enter the name of the SWC in the field Filter and choose Go to get to the underlying SWCV.
7. Once the system retrieves your value, select it and choose Define Prerequisite Software Components.

8. The system displays the dependencies between the prerequisite SWCV (that contains the services to be enhanced) and the dependent EnSWCV.
3. Developing in ESR

3.1. Importing an EnSWCV into ESR

The EnSWCV that we defined previously is ready for implementation in ESR. The following steps describe how to import an EnSWCV from SLD into ESR.

**Step 5: Importing an EnSWCV from SLD to the ESR**


2. Choose the Objects tab.

3. Choose Tools $\rightarrow$ Transfer from System Landscape Directory $\rightarrow$ Import Software Component Versions.
4. Select the EnSWCV from the list and choose Import.

5. The system displays an Import Protocol after it successfully imports the SWCVs.

After Step 5 you see your EnSWCV in the object tree. You also see the folder Basis Objects where all namespaces and their objects from the underlying SWCV are located.
3.2. Creating a Namespace in EnSWCV

You must create a new namespace in the ESR to identify the objects in the SWCV before you develop an enhancement structure in the EnSWCV. To do so, select a unique name such as the URL address of the company, together with some additional information that identifies the application.

**Step 6: Creating a Namespace in EnSWCV**

1. Choose the *Switch Between Display and Edit Modes* pushbutton to go to the change mode in the EnSWCV editor.

2. Insert the new namespace and save it (in the example, [http://xiTest.com/xi/XI](http://xiTest.com/xi/XI) is the new namespace).

3. In the *Change Lists* tab, right click the namespace and choose *Activate*.

4. Choose the *Details* tab to see the EnSWCV that depends on the underlying SWCV.

You can now use the namespace to create enhancements for your Enterprise Service.
3.3. Creating an Enhancement Data Type in the SWCV

Data types that you can use as a basis for future enhancements can be found in the namespaces in *Basis Objects* and in the local namespace. You can enhance complex data types using all of the namespaces from *Basis Objects*. For more information about choosing the right data types, see section 5.2.

Step 7: Creating an Enhancement Data Type in the SWCV

1. Navigate in the EnSWCV to the desired namespace.
2. On the Objects tab, select Data Type Enhancements and choose the Create Object pushbutton.

3. Enter the Name and Description of the data type enhancement and choose Create.

4. On the Enhancement Definition tab, choose the data type to be enhanced.
5. Select the relevant data type from the list that appears. It contains the data types that are located in the namespaces in the underlying SWCV.

6. Choose OK.

7. Choose the Enhancement Definition tab and create the structure of the enhancement data type. For more information about creating elements in enhancement data types, refer to section 5.2.

8. Save the enhancement structure.

9. In the Change Lists tab, right click the object and choose Activate.
10. The system displays a confirmation message after the objects have been successfully activated.
4. Developing in the Backend System

You need to carry out certain steps in the back-end system before you can use the enhanced interface objects from SAP NetWeaver SAP NetWeaver PI.

In the back-end system you generate a proxy structure for the enhancement (see section 4.1). The ABAP system generates an append structure for the service node that is to be enhanced. The system automatically adds the fields from the enhancement to the Web Services Description Language (WSDL) as optional fields.

4.1. Generating an Enhancement Proxy Structure

**Step 8: Generating an Enhancement Proxy Structure**

1. On the SAP Easy Access screen, enter the transaction SPROXY.

2. In the EnSWCV, navigate to the namespace that you created in Step 6 and choose the folder Data Type Enhancement.

3. Double-click the enhancement name and choose Create Proxy. Choose Yes on the dialog box.
4. Enter the Package and the Prefix for your enhancement.

Note: The system needs a prefix to make the enhancement proxy unique within the application. The enhancement structure and its elements contain the prefix as part of the name; in this example, we used the prefix ZTEST. For more information, see SAP Exchange Infrastructure: Guide for Customer Developments and Modifications in the Integration Repository, Release SAP XI 3.0 as listed in Documentation.

5. Generate the enhancement proxy.

6. After the proxy is generated, the system creates an append structure to the enhanced structure as shown in the dialog box below.
7. To check the append structure, double-click the ABAP name under Proxy-Append Structure.

8. The element structure of the append enhancement proxy appears as shown below:
9. To check the enhanced structure, double-click the ABAP name. In this example, you see the enhancement proxy structure ZTESTZTEST as an append structure to the enhanced structure /SAPAPO/SPPKEY FIGURE VALUE.
4.2. BAdI Implementation

After Step 8 you must fill the fields in the enhancement proxy. We provide a BAdI with the following two methods for every service proxy:

- Inbound Processing
- Outbound Processing

For more information about implementing BAdIs, refer to Business Add-Ins (BAdIs) as listed in Documentation.

**Step 9: Implementing BAdIs**

In IMG Customizing, navigate to the BAdI corresponding to your service. Fi shows the BAdIs that belong to the services in Advanced Planning and Optimization (APO) in SCM.

4.3. Enhancing Synchronous Enterprise Services

When enhancing an enterprise service with an enhancement element, there are two possible scenarios:

- The enhancement element is part of the standard business logic of the enterprise service but it is not exposed in the service signature [see Scenario 1].
- The enhancement element is neither part of the standard business logic nor exposed in the service signature [see Scenario 2].

Synchronous enterprise services are all structured the same way and consist of three different message types: an input message, an output message, and a fault message. You can enhance either the input message or the output message. Thus, each of the two scenarios has two variants, depending whether you enhance the input message or the output message.
4.3.1. Scenario 1-Enhancing Service Signature and Mapping to Existing Implementation

In this scenario, the enhancement element is part of the standard business logic of the enterprise service but it is not exposed in the service interface. Thus, you must first enhance the service signature in ESR and then map this new enhancement element from ESR to the corresponding element in the standard business logic. This mapping is done in the enterprise service BAdI, either in the inbound processing BAdI if you enhance the input message, or in the outbound processing BAdI if you enhance the output message.

In this scenario, you perform the following two steps (see figures of Scenarios 1a & 1b):

1. Enhance the enterprise service definition in ESR.
2. In the Enterprise Service BAdI implementation, map the enhanced element defined in ESR with the corresponding element of the standard business logic. Verify the error handling by checking that the value passed by the consumer is correct, if not return an error.

Input Message Enhancement

You want to enhance the input of a synchronous enterprise service. For example, you are using the enterprise service A and you want to enhance it with the element x. The standard implementation of the enterprise service A already "knows" about this element x, that is, A already writes this field in the corresponding database. However, the element x is not defined in the service signature, which means that this field is not part of the request message data type.

To enhance the input of a synchronous enterprise service:

1. Enhance the input message of the service definition in ESR as described in the Figure 6 below.
2. In the inbound BAdI, map the field x from ESR to the field xx from the standard business logic. Verify the error handling by checking that the value passed by the consumer is correct, if not return an error.

The process described above is depicted in the following figure:
Output Message Enhancement

You want to enhance the output of a synchronous enterprise service. For example, you are using the enterprise service Read Customer Basic Data V2 to read the master data of a customer and you want to read the additional element *Train Station*: this is the name of the train station that the customer uses for receiving routine deliveries but may differ from the station that the customer uses for express deliveries. The standard implementation of the enterprise service Read Customer Basic Data V2 already “knows” about the element *Train Station*, which means that Read Customer Basic Data V2 already reads this field from the corresponding database (database KNA1, field BAHNS). However, the element *Train Station* is not defined in the service signature, which means that this field is not part of the output message.

To enhance the output of a synchronous enterprise service:

1. Enhance the output message in the service definition in ESR as described in the figure of Scenario 1b below.

2. In the outbound BAdI, map the field *Train Station* from ESR to the BAHNS field from the standard business logic. Verify the error handling by checking that the value passed by the consumer is correct if not return an error.

*Note*: If there is an error in outbound mapping, it is a program bug in the provider, not a fault of the consumer.

The process described above is depicted in the following figure:

![Figure 8: Scenario 1b: Enhance Service Output](image-url)
4.3.2. Scenario 2-Service Enhancement Requiring Application Enhancement

In this scenario, the enhancement element is neither part of the standard business logic of the enterprise service nor exposed in the service interface. Thus, enhancing the enterprise service requires enhancing the application in addition to the steps described in Scenario 1. To enhance the application, you must use the enhancement concept that the application is providing.

In this scenario, you perform the following three steps (see figures of Scenarios 2a & 2b):

1. Enhance the enterprise service definition in ESR.
2. Enhance the business functionality using the enhancement concept of the corresponding application. (For example, if the application consists of a call to a BAPI, you have to use the extension concept that this BAPI is using, such as user-exits or BAdIs. You also need to extend the BAPI’s signature.) If necessary, enhance the corresponding databases.
3. In the enterprise service BAdI implementation, map the enhanced element defined in ESR with the corresponding element that you added into the application. Verify the error handling by checking that the value passed by the consumer is correct, if not return an error.

Note: As the standard business logic of different enterprise services are implemented differently and/or are using different extension concepts, there is no general recommendation about how to enhance the standard business logic. Depending on the cases, you can either enhance the standard business logic or implement your custom business logic in the enterprise service BAdI itself. However, we do not recommend the second alternative, which should be adopted with caution as it can lead to performance issues.
**Input Message Enhancement**

You want to enhance the input of a synchronous enterprise service. For example, you are using the enterprise service A and you want to enhance it with the element x. For this purpose, you first have to enhance the standard business logic of the enterprise service A so that it writes this field in the corresponding database. Then you follow the same procedure as for Scenario 1a.

The process described above is illustrated in the following figure:

![Diagram](image_url)

**Figure 9:** Scenario 2a: Enhance Service Input and Application
Output Message Enhancement

You want to enhance the output of a synchronous enterprise service. For example, you are using the enterprise service ‘A’ to read some master data and you want to read the additional element y. For this purpose, you first have to enhance the standard implementation of the enterprise service ‘A’ so that it reads this field from the corresponding database. Then you follow the same procedure as for Scenario 1b.

The process described above is illustrated in the following figure:

Figure 10: Scenario 2b: Enhance Service Output and Application
4.4. Enhancing Asynchronous Enterprise Services

Enhancing an asynchronous enterprise service is similar to enhancing a synchronous one. The only difference is how you test the asynchronous enterprise service, as it cannot be tested using Web Service Navigator. However, depending on the services, you can test them either using NetWeaver PI with counter part services or directly test the generated proxy in the back-end system. Otherwise refer to the two scenarios in the enhancement implementation described previously in the first section (Enhancing Synchronous Services).

4.4.1. Example of Asynchronous Enterprise Services

You want to enhance the asynchronous enterprise service Product Activity Notification. This asynchronous inbound message interface transfers the product-related activities of a customer to the suppliers. You are enhancing the node Item → Inventory with the enhancement element Inventory Quantity.

As you can see in the service definition, no type is assigned to the node in the Enterprise Service Repository; however the parent node, Item, does have a type assigned to it. The relationship between Item and Item → Inventory has the cardinality 0.1, which means that there is a maximum of one inventory per item. Thus, you can enhance the node Item with the element Inventory Quantity.
In this example, the enhancement element is neither in the service signature nor it is part of the service standard implementation; and it does not exist in any database of the back-end system. Thus, we are in the same situation as described in Scenario 2a.

In this example, the user must perform the following four steps (see asynchronous service):

1. Enhance the enterprise service definition in ESR.
2. Enhance the database.
3. Enhance the standard implementation of the enterprise service. Since this standard implementation consists of a call to a BAPI, you can enhance the BAPI with the existing BAPI user-exit concept.
4. In the BAdI implementation, map the enhanced element defined in ESR with the corresponding element that you added into the standard implementation.

The process described above is illustrated in the following figure:

![Figure 11: Example of Asynchronous Service Enhancement](image)
5. Appendix

5.1. Relationship between Products, Product Version, SWCs, and SWCVs in SLD

The relationship between the product and the SWC in SLD is illustrated in the following figure:

![Diagram](image-url)

**Figure 12**: Relationship Between Product and SWCV in SLD

It can be better understood by using the following example: the product SAP Supply Chain Management (SAP SCM) has multiple component versions such as SAP SCM 5.0 and SAP SCM 5.1.

A SWC is defined as a reusable module of a product and can be individually upgraded. For more information, see SAP Exchange Infrastructure: Guide for Customer Developments and Modifications in the Integration Repository, Release SAP XI 3.0 listed in Documentation.

A SWCV is defined as a particular version of a SWC. For example, SAP BW 2.1C and SAP Basis 4.6C are versions of SAP BW and SAP Basis.

A software unit describes a relationship between a software product and a SWC.
Example

If you plan an enhancement for the software component version SAP SCM 5.1, you can name the product Enhancement_Product_SAP_SCM and the SWC Enhancement_SWC_SAP_SCM. If you assign version numbers to the product and to the SWC, the names for the product version and the SWCV are automatically assigned by the system.

5.2. Creating an Enhancement Structure in EnSWCV

When you have identified the data type to be enhanced, you can then decide the structure of the enhancement and choose the data type to assign to the elements in the enhancement structure.

Example

Refer to the example from the Step 7. You decided to enhance the data type SPPKeyFigureValue. This data type contains four fields to transfer different numbers. For example, the field CriticalProductNumberValue contains the numbers of products with critical shortage status in the Service Part Supply Plan. In this service we communicate the numbers of products, for which special transport regulations exist.

In ESR, create an enhancement with the name z_test to enhance the node SPPKeyFigureValue. Create the enhancement structure in the enhancement definition and insert the elements or attributes in the enhancement structure (in the example ZcustomerNumberValue). Assign a type such as NumberValue to the element.
We recommend the following three ways to assign a data type to an element. They are listed in order of preference:

1. Use a data type from an underlying SWCV (from all namespaces in this SWCV)
2. Use a data type from the enhancement namespace (in the example http://xiTest.com/xi/XI)
3. Use a built-in data type integrated in ESR

5.2.1. Using a Data Type from an Underlying SWCV

To find data types in the namespace where the service operation was originally located, navigate to EnSWCV → Basis Objects → Namespace → Data Types. Here you see a list of all the data types that are used in this service operation. For example, to assign a type to the element ZcustomerNumberValue, you can use the data type NumberValue or any other suitable data type that you find in the underlying SWCV.
5.2.2. Using a Data Type from the Enhancement Namespace

1. To create the data type in the namespace in EnSWCV, navigate to EnSWCV → Namespace → Interface Objects → Data Types. On the menu bar, choose Create Objects.

2. Enter the Name and Description of the data type and choose Create. Double-click Details and enter the length of the data type from 0 to 999999999.

3. After saving, activate the data type within the Changes Lists tab.
4. After activation, the element \( z\text{\_customer\_numbervalue} \) appears as a data type in the enhancement namespace.

5.2.3. Using a Built-In Data Type Integrated in ESR

The elements of the enhancements structure can also be assigned a type using a built-in data type. The World Wide Web Consortium has defined 19 built-in data types (http://www.w3.org); SAP has also defined some data types such as \texttt{xsd:integer} and \texttt{xsd:short} (Figure 2).

Use the search help in the Type column to open the list of data types from all the namespaces of EnSWCV and the underlying SWCV. In the example, to use the element \texttt{ZcustomerNumberValue}, we use the reference data type from the namespace of the underlying SWCV. The data type \texttt{NumberValue}, from one of the namespaces in the underlying SWCV, is selected for \texttt{ZcustomerNumberValue}. The prefix 'p1' indicates that the data type is selected from a namespace other than the namespace where the enhancement is developed.
After you have created the enhancement structure, save the enhancement and activate it within the Changes List tab.

5.3. Impact of Update or Upgrade of an Underlying SWCV on an Enhancement

EnSWCVs remain unaffected by updates of underlying SWCVs. Thus, the enhancements developed in SAP NetWeaver PI remain stable after updates.

5.4. Documentation

1. Enterprise Services for SAP, ERP 6.0
   

   
   [Link](https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/00cc32a1-d311-2a10-f39b-aa1cfa7b7d78)

3. Business Add-Ins (BAdIs)
   
   [Link](http://help.sap.com/saphelp_nw70/helpdata/en/99/9cdd4013e8531e1000000a1550b0/frameset.htm)

4. Service Provisioning Handbook
   
   [Link](https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/6d19c8ee-0c01-0010-619d-92af980436d7)

5. Customer enhancement and modification of BAPIs
   
   [Link](http://help.sap.com/saphelp_nw70/helpdata/EN/7e/5e141a4a1611d1894c0000e829fbbd/frameset.htm)
## 5.5. Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAdI</td>
<td>Business Add-In</td>
</tr>
<tr>
<td>BAPI</td>
<td>Business Application Programming Interface</td>
</tr>
<tr>
<td>DDIC</td>
<td>Data Dictionary</td>
</tr>
<tr>
<td>ESR</td>
<td>Enterprise Service Repository</td>
</tr>
<tr>
<td>EnSWCV</td>
<td>Enhancement Software Component Version</td>
</tr>
<tr>
<td>IMG</td>
<td>Implementation Guide</td>
</tr>
<tr>
<td>PI</td>
<td>SAP Process Integration</td>
</tr>
<tr>
<td>SCM</td>
<td>Supply Chain Management</td>
</tr>
<tr>
<td>SLD</td>
<td>System Landscape Directory</td>
</tr>
<tr>
<td>SWC</td>
<td>Software Component</td>
</tr>
<tr>
<td>SWCV</td>
<td>Software Component Version</td>
</tr>
<tr>
<td>XI</td>
<td>SAP Exchange Infrastructure</td>
</tr>
</tbody>
</table>