SAP NetWeaver Process Integration
Best Practices: Design

Applicable Releases:
SAP NetWeaver Exchange Infrastructure 3.00
SAP NetWeaver Process Integration 7.00
SAP NetWeaver Process Integration 7.1x

Topic Area:
SOA Middleware

Capability:
Service Bus

Version 1.0
December 2008
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>First official release of this guide</td>
</tr>
</tbody>
</table>
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example Text</td>
<td>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</td>
</tr>
<tr>
<td>Example text</td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles</td>
</tr>
<tr>
<td>Example text</td>
<td>File and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td>Example text</td>
<td>User entry texts. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td>&lt;Example text&gt;</td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td>EXAMPLE TEXT</td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚨</td>
<td>Caution</td>
</tr>
<tr>
<td>📝</td>
<td>Note or Important</td>
</tr>
<tr>
<td>📠</td>
<td>Example</td>
</tr>
<tr>
<td>🕹️</td>
<td>Recommendation or Tip</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **Background Information** .............................................................. 1

2. **Service Structure Definition with Interface Objects** ....................... 2
   2.1 Service Interfaces and Interface Pattern ....................................... 2
   2.1.1 Interface Pattern: Stateless ....................................................... 2
   2.1.2 Interface Pattern: Stateless XI 3.0 Compatible ......................... 3
   2.1.3 Best Practices: Interface Pattern Stateless vs. Stateless XI 3.0 Compatible .. 3
   2.2 Data Type Definition and Usage .................................................. 4
   2.2.1 Pre-delivered SAP data types ................................................... 4
   2.2.2 Usage of data types ............................................................... 5
   2.2.3 Data Type Enhancements ....................................................... 9

3. **Mapping** .................................................................................... 13
   3.1 Mapping Types – Choosing the Right One ..................................... 13
   3.2 When to Look At Alternative Mapping Types ................................ 16
   3.3 Taking Advantage of New Mapping Features & Enhancements in PI 7.1 ........ 17
   3.4 Data Lookup Considerations ..................................................... 18
      3.4.1 Value Mapping vs. Mapping Lookup ..................................... 20

4. **Integration Processes (ccBPM)** .................................................. 22
   4.1 Using Integration Processes Correctly .......................................... 22
      4.1.1 Checklists for Integration Processes ................................. 22
      4.1.2 Typical Scenarios ........................................................... 22
   4.2 Integration Process Best Practices .............................................. 23
      4.2.1 Assess Suitability ............................................................ 23
      4.2.2 Not an Answer for Everything .......................................... 23
      4.2.3 Take Advantage of New BPE Runtime Features .................. 25

5. **Additional Design Topics** .......................................................... 31
   5.1 Usage of Local Software Component Version .............................. 31
   5.2 When to Use Outside-In and Inside-Out Approach for Defining Services .... 31

6. **Appendix** .................................................................................. 32
1. Background Information

This guide is part of a how-to guide series providing best practices and guidelines for PI and SOA processes. The current guide deals with best practices in terms of design practices concentrating on the use of objects within the Enterprise Services Repository (ES Repository). Where information is available otherwise, the guide will refer to the same rather than to repeat it.
2. Service Structure Definition with Interface Objects

2.1 Service Interfaces and Interface Pattern

Starting with SAP NetWeaver Process Integration 7.1, a Service Interface is a platform and language independent design time representation of a service. It describes one or more operations to be implemented in an application system and is used in mediated scenarios via the Integration Server or Point-to-Point scenarios via the Web Service Runtime. Service interfaces are the evolution of Message Interfaces that are used in SAP NetWeaver PI 7.0 and XI 3.0.

Interface Pattern is a new attribute of a service interface that describes the type of communication that is to be executed on the message.

When defining a service interface, four potential interface patterns are available to choose from: Stateless, Stateless (XI 3.0 Compatible), TU&C/C, and Stateful. For SAP NetWeaver PI, only Stateless and Stateless (XI 3.0 Compatible) are relevant for use and thus, only those will be discussed in the following.

2.1.1 Interface Pattern: Stateless

Key Attributes

- Mediated web services communication via WS Adapter.
- Direct Connection communication, i.e., point-to-point (P2P), via the Web Service Runtime

Note
Consumers & providers must have at least ABAP 7.0 (for async >= SP14), ABAP 7.1x in order to enable direct connection via the web service runtime.

- Single or multiple operations can be defined within a service interface
- Asynchronous and synchronous operations can be combined within the same service interface
- Operation Mode “Synchronous”
  - Simple web service call (no reliable messaging)
  - Supported adapters: WS-RM, SOAP, ABAP SOAP entry
• For local processing in Advanced Adapter Engine (AAE), i.e., the so-called integrated configuration object, SOAP adapter has to be used

• Operation Mode “Asynchronous”
  o WS-RM protocol is used
  o Supported adapter: WS-RM only
  o QoS EOIO only, no EOIO across service interfaces
  o No local processing in AAE supported

2.1.2 Interface Pattern: Stateless XI 3.0 Compatible

Key Attributes

• Limited to exactly one operation
• Name of the service interface and operation must be identical
• Mediated scenarios via proxies or technical adapters to connect to legacy or 3rd party systems
• XI 3.0 protocol not used in P2P case, communication via Web Service runtime
• Consumers & providers: ABAP 7.1, 7.0, 6.40, 6.20 (latter with plug-in), SAP JEE5, J2EE 1.3
• Direct Connection not supported

2.1.3 Best Practices: Interface Pattern

Stateless vs. Stateless XI 3.0 Compatible

Stateless

• This pattern can be seen as the default pattern for developing new services in ES Repository
• Recommended for newly designed services with multiple operations or mixed modes
• Allows for switching from P2P to a mediated scenario (and vice versa) by configuration only, and without changing the programming model

Stateless XI 3.0 Compatible

• This interface pattern is used for all existing XI 3.0 / PI 7.0 message interfaces that are migrated to service interfaces in PI 7.1
• Recommended pattern for XI/PI specific scenarios that use the common “technical adapters” such as File, JDBC, JMS, etc. (basically all the adapters except for the WS adapter)

More Information:

• SAP Help Portal: Service Interface and Interface Pattern
• SAP Help Portal: Direct Connection
2.2 Data Type Definition and Usage

2.2.1 Pre-delivered SAP data types

Introduction

SAP delivers descriptions of Global Data Types (GDTs) as Enterprise Services Repository (ESR) content to customers. Customers can create their own data types in the ES Repository and use SAP GDTs.

- Two ways of modeling data types are supported: Classical XSD style and new CCTS (Core Component Technical Specification) style.
- Data types are classified as free-style (classical), core (atoms of CCTS modeling), or aggregate (complex CCTS).
- Standard fault data types are no longer created automatically for every namespace (only when fault message type is created).
- Additional functions in data type editor provide better support for structural changes (e.g., moving subtrees).

More Information:
SAP Help Portal: Global Data Types
According to this, we distinguish between SAP Core Data Type and SAP Global Data Type. The SAP Core Data Type as well as the W3C Type (XSD type) does not have any business semantics. The SAP GDT has business semantics, it actually can be based on either a core or an aggregated DT.

Free-style data types are directly based on the primitive data types, e.g., xsd:decimal. They do not need any further parameters to define themselves.

More Information:

- SAP Help Portal: Core Data Types and Aggregated Data Types

**Recommendation**

**When to use free, core and aggregated data types**

Core data types are based on CCTS specification. Aggregated data types have business semantics, are based on CDTs, and are the basis for application-specific data types SAP-wide.

Customers who adhere to the CCTS specification or have business scenarios dealing with SAP backend systems, should try and reuse the existing core and aggregated data types delivered by SAP. However, customers who wish to use their own schema for defining the data types, free-style data type is the preferred approach.

While transferring the business scenarios from XI 3.0/PI 7.0 to PI 7.1, all the data types in the design objects will be transferred as free-style data types by default.

While creating new objects in PI 7.1x, it is recommended to reuse the core and aggregated global data types provided by SAP.

The detailed list of SAP delivered data types is available on SAP Developer Network (SDN) at ES Workplace.

https://www.sdn.sap.com/irj/sdn/esworkplace

### 2.2.2 Usage of data types

**Free-style data type**

Free-style data types can be created by using the XSD types, reusing core data types and free-style data types.

*In the example below we are creating a Student_Marks_DT as a free style data type. It uses both core data type and XSD types.*

Following steps are performed:

1. Define Roll_No by referring a core data type Referenceld. This core data type is of type numeric with maxlength 10.
2. Define parameters marks1 and marks2 as XSD types.
Free style data type using a core data type:

Core data types

Core data types are based on representation terms which are ISO 15000 5 (ebCCTS) compliant. SAP delivers core data types with the same name as representation terms. Applying the standard methodology simplifies the implementation of cross-company processes.

CDTs are intermediate data types, and are a level above primitive data types like decimal, string, etc. They are defined by representation terms such as Amount, Identifier, etc. They are themselves not yet application-specific and are therefore referred to as context-free.

While creating core data type based on any given representation term, we get an option to either use a default XSD type or reuse an existing core data types.

For using the standard XSD types we select Type as ‘XSD Type’

For reusing already existing core data type we select Type as ‘Data Type’

**Prerequisite**

Define a dependency on the SWC SAP Global to reuse any of the standard core data types shipped by SAP.

**Example on defining ‘Type’ as ‘XSD Type’**

To create EmployeeId_DT, we can use Identifier representation of core data type. Identifier is unqualified (i.e., has no attributes) CDT. It has no details with respect to min length and max length. We can define the qualifiers and specify a pattern while reusing the data type as per the specification required for defining the EmployeeId.
Example on defining ‘Type’ as ‘Data Type’

Another means is to develop core data types from user defined CDTs. This is generally helpful for creating another layer of abstraction.

There are core data types which have detailed list of attributes. For example: if we use a ‘Code’ representation term then it has a detailed attribute list which can be filled as per the requirement by the user.

In a business scenario, we might need a structure which could be common to multiple data types.

As first step we check the already available CDTs, find the closest match, build on top of it and then reuse it.

In our example, we create a data type called “ReferenceId” using the Identifier representation term. Since Identifier is an unqualified CDT, we need to provide qualifiers to develop a ReferenceId, such as, adding restrictions in the details column. Once created and activated, ReferenceId can be used as a substructure to multiple data types which could be either core or aggregated.

EmployeeId_DT core data type refers ReferenceId thus creating one more level of abstraction.

Note

Core data types can be qualified (i.e., has attributes defined) as well as unqualified (i.e., has no attributes defined).

Note

The representation term for the core data type does not yet define any business semantics with a specific reference to business processes, but rather semantics for the more precise characterization of the type. For example, the characterization of a data type with the Amount representation term defines that the value of such a type must also be specified further by means of a currency. If you model such a data type freely as an XSD type (for example xsd:decimal), this obligatory characterization is missing.

Each amount needs a currency, if you model it with a free-style data type this information/link is missing. Such cases core data type is recommended!

More Information:
SAP Help Portal: Developing Core Data Types
Aggregated data type:
Aggregated data types have business semantics, are based on CDTs, and are the basis for application-specific data types SAP-wide.

Please refer to link below for detailed understanding: [Introduction to Global Data Types in SAP NetWeaver PI 7.1 (preview)]

Prerequisite
To reuse already existing core or aggregated data type, we need to define dependency on the original software component version. All standard Global data types are available in SAP Global software component.

Example of developing an aggregated data type reusing the core data type and aggregated data type:

1. Define a new aggregated data type

2. To define field Emp_Id we reuse user defined core data type Employeeld_DT, for Emp_Name we use aggregated data type Name and for Emp_Address we use AddressElements.

Note
- References to free-style data types are not permitted.
- In case of transfer of objects from one system to another, all the underlying software component versions should also be transferred.
**Recommendation**

*Depending on the proxy type, the following tasks must be done before you generate a proxy:*

All objects must be modeled in the ES Repository — or be made available as external WSDL descriptions.

During modeling of proxy in the ES repository, there are scenarios where we need to reuse an object from underlying SWCV to the based-on SWCV. It is recommended to create the new dependent object of based-on SWCV under a different namespace because the namespace must be unique. Before generating the proxy for the dependent object, the proxy for the referenced object should be created explicitly.

In case the new dependent object belonging to the based-on SWCV is created under the same namespace as the underlying SWCV, the proxy for the referenced object will be automatically generated while generating the proxy for the new object. It happens with some warnings. However, it is not a recommended approach.

### 2.2.3 Data Type Enhancements

Aggregated data types can be normally enhanced as was the case with data types in PI 7.0.

In the below example we are trying to enhance the EmployeeDetails_DT aggregated data type. EmployeeDetails_DT is a data type defined in SWCV I029930_PI 7.1 of sap

Following steps are performed:

1. As a first step we need to identify the node on which the enhancement should to be done.
2. Enhanced data type is created in the separate software component version I029930_PI7.11 of sap.
3. For SWCV I029930_PI7.11, dependency is defined on the I029930_PI 7.1.
4. EmployeeDetails_ehDT is the enhanced data type defined in I029930_PI7.11. This enhanced structure is added as a node to the original aggregated data type.
5. To use the enhanced data type and generate a proxy, a message type should be created in the based-on software component version, i.e., I029930_PI7.11.

6. Develop a message type in the I029930_PI7.11. This should point to the data type from the Basis objects I029930_P17.1 → EmployeeDetails_DT

---

Note

Only under the Basis Objects I029930_PI7.1, we can see the data type with the enhanced version and not in the original software component, i.e., with the additional node.
7. Create a service interface using the new message type, i.e., EmployeeDetails_ehMT

8. Generate the proxy at the backend for the service interface containing the enhanced data type.

For details on enhancement, please refer to the enhancement guide: [Enterprise Services Enhancement Guide](#)

**Note**

The enhanced data type can be created in any software component with any namespace provided a proper dependency is defined with the original software component version.

**Recommendation**

Enhanced data type is generally for customer implementation. It is recommended to define data type enhancements in separate software component version and define a dependency on the software component from where the original data type is taken.

**Limitation**

Core data types cannot be enhanced.
Note (Observations)

- If the original data type is modified and activated, all the objects referencing this data type will also get updated.

- Under dependent SWCV, if original basis data type is changed, these modifications are stored locally to the based-on SWCV. Such changes do not get reflected in the original SWCV. Later sometime, if the same data type gets changed in the original SWCV, the modification and activation of original data type does not get reflected automatically in based-on SWC where it was changed locally. In all other based-on SWCVs, modification of the original data type does get reflected.

- Due to changes in the data type both in local as well as original SWCV, a version conflict appears in the data type at based-on SWCV. User is given a choice to either choose local changes made in local SWCV or those made in the underlying SWCV.

- There is no merge option available between the local modifications and the modifications done on the original object in underlying software component.
3. Mapping

Mapping messages from one format/structure to another is a fundamental feature for any middleware application in A2A and B2B scenarios. SAP NetWeaver Process Integration offers a wide variety of mapping program types from which customers can choose from:

- Message Mapping (Graphical Mapping Tool)
- XSLT Mapping (Java Engine)
- Java Mapping
- ABAP Workbench Mapping Programs
  - ABAP Mapping
  - XSLT Mapping (ABAP Engine)

3.1 Mapping Types – Choosing the Right One

Providing support for a variety of mapping program types offers obvious advantages in terms of overall flexibility considering that each customer faces a unique set of circumstances. Still, it helps to evaluate the different mapping types against a common set of criteria in order to compare their relative strengths and weaknesses. Below are some of the main considerations to look at when assessing the PI mapping types.

Mapping Type Criteria

**Features and Functions:** underlying mapping technology must be able to provide enough features and functions such that mappings are implemented efficiently and effectively. It should also provide support for complex mappings.

**Performance:** considers the execution time as well as system resource usage, especially for medium to large sized messages.

**Standards and Portability:** mapping programs based on open standards allows for greater portability.

**Usability:** considers the ease of use and user-friendliness of the underlying mapping technology with emphasis on minimizing the need for a specialized skill set (e.g. Java, XSLT, ABAP). For instance, how easily and quickly an integration developer/consultant without a special skill set can implement a moderately complex mapping.

**Software Logistics:** maintenance and change management considerations such as versioning and transports.

**Total Cost of Ownership (TCO):** an overall consideration of the mapping type criteria and assessment of the general cost ramifications of using a particular mapping program type.
Mapping Type Comparison

The following table compares the entire set of SAP NetWeaver PI mapping program types available based on the criteria outlined above. A brief explanation of the ratings for each mapping type is provided following the table below.

⚠️ CAUTION
The table below is a general guideline to assist SAP NetWeaver PI architects and developers in making a strategic decision or following a strategic direction in terms of SAP NetWeaver PI mapping technology. It is highly conceivable that individual mapping requirements can contradict the “ratings” provided below as any given mapping requirement can be met in a countless number of ways within a given mapping type.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Message Mapping</th>
<th>Java</th>
<th>XSLT</th>
<th>ABAP</th>
<th>XSLT (ABAP Engine)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features and Functions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standards/Portability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Usability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Logistics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Legend

- **Recommended**
- **Acceptable**
- **Not Recommended**

*** Assumes highly experienced developer

Message Mapping

Message mappings are created using a user-friendly, graphical mapping tool that allows for new users to implement and test mappings quickly and effectively. It offers a rich set of features, standard functions, and the option to create custom user-defined functions to support the most complex of mappings. New features and enhancements for message mappings are steadily produced on an ongoing basis. This native mapping tool for PI is integrated in the Enterprise Services Builder, thus inheriting the software logistics and versioning aspects that the Enterprise Services Builder provides. Although well written Java programs from experienced Java developers may possibly offer the best...
SAP NetWeaver Process Integration Best Practices: Design

Performance, message mapping performance usually produces a linear relationship between time and message size and usually surpasses that of XSLT and the ABAP-based mapping programs.

**Recommendation**
SAP recommends the use of Message Mappings as the default mapping type and to consider the use of other mapping types in exceptional cases where extensive advantages are clearly evident.

**Java Mapping**
Handwritten Java programs by experienced Java developers are usually quicker than generated Java source code from message mappings. Experience has also shown that Java-based mapping programs are quicker than ABAP-based mapping programs. The Java programming language is an open standard and provides a rich set of functions and features constantly evolving and improving with new JDK releases. It is possible that experienced Java developers can turn around a complex mapping requirement more efficiently and even with better mapping performance. The main downside is that Java is still a specialized skill set that companies may not have or have enough of. Even if the skill set is available, if an organization decides to make a strategic change in direction away from Java or other specialized skill sets, the cost of maintenance and possible subsequent migration of mappings can become high and can increase TCO.

**Note**
Performance of Java and ABAP mappings in particular is highly dependent on the experience and skill level of the respective developers.

**XSLT Mapping**
Like the other mapping types, XSLT also provides a rich set of features and functions based on open standards that can be used to efficiently address complex mapping requirements. However, one of the main concerns with XSLT is its performance in mapping large messages. Some preliminary tests have shown that execution time can decrease dramatically as message size increases. However, these tests were based on the transformation engine available on the SAP XML Toolkit. From PI 7.1, the transformation engine in JDK 5 is used for XSLT mappings and is claimed to provide quicker execution. However, new data for large message performance using JDK 5 is currently unavailable. Also, as with Java, XSLT is a specialized skill set that organizations may not have or have enough of. If an organization decides to make a strategic change in direction away from XSLT, the cost of maintenance and possible subsequent migration of mappings can become high and can increase TCO.

**ABAP Workbench Mapping Programs**
There are powerful programming features available in the ABAP workbench to address complex mappings by experienced ABAP developers. As with Java and XSLT, the drawbacks of requiring a specialized skill set also applies to ABAP. In terms of performance for ABAP XSLT, execution time increases dramatically as message size increases. In mediated scenarios, ABAP based mappings do not require a JCo call to the java mapping runtime which helps with overall mapping performance. In addition, SAP does not ship ABAP mapping programs and there is no mechanism for shipping mapping programs with SAP applications and importing them into the Integration Server. Mappings must be created directly on the Integration Server or transported there and cannot be imported into the ES Repository. In terms of versioning, ABAP mappings and XSLT mappings (ABAP Engine) can only exist in one active version on the Integration Server. In contrast, the Java, XSLT, and message
mappings that are executed on the J2EE Engine can be used in multiple versions in parallel. Finally, there is no access to value mapping tables from ABAP based mappings.

Note
Performance of Java and ABAP mappings in particular is highly dependent on the experience and skill level of the respective developers.

3.2 When to Look At Alternative Mapping Types

Although SAP recommends the use of message mappings to address mapping requirements when using SAP NetWeaver PI, there will inevitably be a specific mapping requirement or a unique set of company circumstances that warrant the use of the alternative mapping types. The following circumstances are some examples where it might make sense to use an alternative mapping type.

- Migrating XSLT or Java Mapping programs from a previous integration project

Note
Regarding XSLT mapping programs, note that SAP XML Toolkit is being phased out in favor of the XSLT engine that is native to JDK 5. Customers should use JDK 5 for all new XSLT mapping programs (default option) and test existing XSLT mappings with JDK 5 and make any modifications if necessary. The option to select the runtime (SAP XML Toolkit or JDK 5) is available in the operation mapping. SAP XML Toolkit will continue to be supported in the next release after 7.1, but will not be supported in any subsequent releases thereafter.

- Experienced Java, XSLT, and ABAP developers in house (and indications remain this way for the foreseeable future)

- Highly complex and/or unique set of mapping requirements that may be better addressed using another mapping technology. Examples include the following:
  - Source or target message is a non-XML file (e.g. a text file)
  - Mapping requirements not conducive for message mappings. For example, many of the message mapping functions are only applicable within a given queue or context of an XML message. For instance, a requirement to sort elements of a message or store and manipulate various elements/subtrees of a document may require very complex user defined functions. In the end, it may be more cost effective to address the mapping requirements using another mapping type (e.g. Java).

- Performance is the highest priority and critical to the business. In this case, a Java mapping by an experienced Java developer, for example, may be able to produce the mapping program with the shortest execution time.
3.3 Taking Advantage of New Mapping Features & Enhancements in PI 7.1

There are many new mapping features and enhancements available in the SAP NetWeaver PI 7.1 release. We highlight a couple of them below. For a more comprehensive look at all the main features, please check the SDN blog series “SAP NetWeaver Process Integration 7.1: Mapping Enhancements Blog Series”.

Parameters in Mapping Programs

Parameters can be used to transfer values to a mapping program at configuration time thereby increasing the number of possible applications for a mapping program. For example, you could first define a separator for a concatenation function in the Integration Directory. In this way, mapping programs can be applied flexibly. All mapping program parameters are known as part of the signature of the mapping program.

Key benefits of parameterized mapping programs:

- Reuse of mappings in multiple Interface Determinations with varying parameters
- Set channel for mapping lookups at configuration time
- Payload information can be written to simple typed containers of an Integration Process

Function Libraries

Mapping developers can use their own user-defined functions in message mappings in addition to the standard functions. The ES Builder saves these functions in a local function library of the message mapping. They are only visible in this message mapping (in the data-flow editor for defining target-field mappings). You generally use user-defined functions to add your own simple or complex Java functions to the functions of the message mapping.
A function library is a separate object type in the Enterprise Services Repository (ES Repository) and enables mapping developers to use user-defined functions across message mappings. You can use user-defined functions from a function library in message mappings and in mapping templates.

The main benefits of function libraries are:

- Enhanced portability of user-defined functions by using function libraries
- Reuse of user-defined functions across different mappings
- Reuse of user-defined functions across different SWCV via dependencies

### 3.4 Data Lookup Considerations

Some integration scenarios require additional data from application systems to enrich the message contents or to simply translate values from one representation to another according to different application or organizational contexts. Such requirements can be addressed in SAP NetWeaver Process Integration using two fundamental mechanisms: **Value Mapping and Mapping Lookups**.

#### Value Mapping

Value mapping is strictly used for translating a value from one representation to another according to different application or organizational contexts.

**Example**

The person Mr. Smith has a particular customer number as a customer of mail order company A and has a particular employee number as an employee of company B. Both numbers refer to the same person.
The mapping rules for different values are saved in a value mapping table where they can be accessed by the Integration Server at runtime. There are three options for creating and editing value mapping tables:

- **Integration Directory UI**: Allows for direct manual input using the user interface of the Integration Directory using two different value mapping views: *value mapping group* and *value mapping agencies*. More information on this option and its two views can be found on the SAP help portal: Displaying/Editing the Value-Mapping Table.

- **Value Mapping Replication for Mass Data**: If the value mapping data is stored in external tables, this data can be replicated to the runtime cache (on the Integration Server) by using special service interfaces (value-mapping replication interfaces). These interfaces allow the implementation of both synchronous and asynchronous replication scenarios. More information on this option can be found on the SAP help portal: Value Mapping Replication for Mass Data.

  **Note**
  
  You cannot display the replicated value mapping data in the Integration Directory user interface since the original runtime data is stored in the application from which the data is copied. However, you can display the replicated value mapping data in Cache Monitoring of the Runtime Workbench.

- **Integration Directory API**: Value mappings and other integration directory objects can be created, edited, and activated in mass using an application programming interface (API). The programming interface consists of web services containing all the necessary operations to create and edit configuration objects and which can be called using a valid client application. In addition, objects created using the directory API can also be manually edited using the “normal” directory user interface if applicable. More information on this can be found on the SAP help portal: Integration Directory Programming Interface.

### Mapping Lookups

By using a mapping lookup, mapping programs can call functions from other application systems while a mapping program on the Integration Server is being executed. This may be necessary for the following reasons:

- To read from application system data in the mapping program
- To call existing mapping routines in the application system

The mapping runtime has a lookup API for calls to application systems. It supports access using the RFC, JDBC, and SOAP adapters. In addition, starting with SAP NetWeaver PI 7.1, RFC and JDBC lookups are supported *graphically* which adds great value by eliminating the need for any custom development as required in previous releases. For more information on mapping lookups, check the SAP help portal: Adding Lookups to Mapping Programs.

  **Recommendation**

  Message mappings using mapping lookups should be designed such that the fewest number of lookup calls are executed. For example, place the lookup call in the highest possible node of the target message and store retrieved data in the mapping *GlobalContainer* object using a user defined function so that the number of lookups made is greatly reduced. This will lead to a significant increase in performance.

In the following, we compare the two mechanisms (value mapping and mapping lookup) and evaluate which one is best suited for a given integration lookup requirement.
3.4.1 Value Mapping vs. Mapping Lookup

In general, comparing Mapping Lookups to Value Mapping (in all its maintainable variations) is somewhat like comparing apples to oranges. They are similar in the ends sought, but the means at which they provide the data and the scope of input and output values vary greatly between the two mechanisms. When trying to determine which mechanism is best suited for a particular data lookup requirement, some of the main factors to consider are the following:

- **Amount of Data**: Is there a small or large amount of data to maintain?
- **Frequency of Change**: Is the data static or dynamic?
- **Location of Data**: Is the data externally maintained?
- **Input/Output Value Ratio**: Is the input/output value ratio 1/1?
- **Availability of Value Mapping Function**: Does the value-mapping function already exist in the backend or can it be easily implemented?
- **Backend Application Type**: Is the backend an SAP system, database system, or application that provides web services?

### When to Use Value Mapping vs. Mapping Lookup

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Conditions for Use:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value Mapping (Integration Directory UI)</td>
<td>• Dataset is small, static, not externally maintained.</td>
</tr>
<tr>
<td></td>
<td>• Input/Output value ratio is 1/1.</td>
</tr>
<tr>
<td>Value Mapping Replication for Mass Data</td>
<td>• Dataset is large, dynamic, and externally maintained.</td>
</tr>
<tr>
<td></td>
<td>• Input/Output value ratio is 1/1.</td>
</tr>
<tr>
<td>Value Mapping (Integration Directory API)</td>
<td>• Dataset is large, dynamic, and externally maintained.</td>
</tr>
<tr>
<td></td>
<td>• Input/Output value ratio is 1/1.</td>
</tr>
<tr>
<td></td>
<td>• Custom client application can be developed with moderate effort.</td>
</tr>
<tr>
<td>Mapping Lookup</td>
<td>• Dataset is large, dynamic, and externally maintained.</td>
</tr>
<tr>
<td></td>
<td>• Input/Output value ratio is n/n.</td>
</tr>
<tr>
<td></td>
<td>• A value mapping function is available in the backend or can easily be implemented.</td>
</tr>
<tr>
<td></td>
<td>• Backend is an SAP system (RFC), database system (JDBC), or web services provider (SOAP).</td>
</tr>
</tbody>
</table>

Notice that some conditions restrict the use of one mechanism versus the other. For example, if retrieving the values requires several input values, a mapping lookup may be the only possible option for the data lookup. Or if the dataset is small and not consistently maintained in an external table,
using value mapping (Integration Directory UI) may be the only possible option. The following recommendations apply when conditions for using both mapping lookup and value mapping are met.

**Recommendation**

In general, when conditions for both value mapping (replication or directory API) and mapping lookup apply, use mapping lookup to avoid redundancies/double maintenance as long as the amount of lookup calls during the mapping execution is kept to a minimum. If too many lookup calls are made during the mapping execution, this can adversely affect system performance and in this case, value mapping would be recommended.

In addition, if value mapping is determined to be more suitable, the recommended value mapping option would be value mapping (integration directory API) due to its advantages such as the possibility to manually maintain objects created via the directory API and also the possibility for it to fit into a more global API strategy.
4. Integration Processes (ccBPM)

An Integration Process can add great value to SAP NetWeaver PI integration scenarios by automating the execution of message-based service orchestration across system and/or application boundaries — thus the reason integration processes are also referred to as cross-component BPM (ccBPM). Integration processes further enhance service orchestration and standard Integration Server mediation services with stateful processing of messages by preserving the status of an integration process on the Integration Server.

Note
The term “stateful” as used here is not to be confused with the interface pattern “stateful” which is another attribute of service interfaces in the ES Repository.

4.1 Using Integration Processes Correctly

Due to its powerful features and the relative elegance and visibility in using graphical process models when designing integration solutions (especially for complex integration scenarios), there is a natural inclination and appeal in broadly applying the use of integration processes across a wide range of integration requirements. However, when it comes to integration processes, a creative and syntactically correct design of an integration process “on paper” does not always translate into proper use, and much less, into efficient use in terms of runtime performance.

Integration processes are executed on the Integration Server at runtime by the Business Process Engine (BPE) and can potentially consume a lot of resources. Since the Integration Server is the central resource for message exchange, it is vital to ensure that it is not overloaded; otherwise, this can lead to bottlenecks or performance problems. See Note 857530 (BPE: Composite Note Regarding Performance) for more information regarding performance information in the BPE.

4.1.1 Checklists for Integration Processes

SAP has produced a series of “checklists” in order to assist customers recognize which scenarios integration processes should ideally be used for:

- Checklist: Making Correct Use of Integration Processes
- Checklist: Making Correct Use of Correlations
- Checklist: Making Correct Use of Dynamic Processing
- Checklist: Making Correct Use of Mappings

All the checklists above can be found on the SAP help portal: Checklists for Integration Processes

4.1.2 Typical Scenarios

To further guide customers in properly applying the use of integration processes, SAP has delivered examples of integration patterns and use cases comprising of the typical process step combinations used in such patterns. They can be found in the ES Repository under the software component SAP Basis → SAP Basis <release number> and namespace http://sap.com/xi/XI/System/Patterns. These examples are actual integration processes that can be used as templates and inserted into customer integration processes. Detailed descriptions of the delivered examples can be found on the SAP help portal: Examples and Usage Cases
4.2 Integration Process Best Practices

The following list of recommendations can be applied during the design phase when exploring the use of integration processes for a given set of integration scenarios. Though the subsequent sections provide some cautionary points when working with ccBPM, it is worth noting that SAP NetWeaver Process Integration has made great strides in addressing some of the performance issues originally raised when integration processes were initially offered in XI 3.0. Section 4.2.3 is a good validation of this. But as with any executable process, proper and efficient design of an integration process in the context of a middleware application can only come with a better understanding of the entire middleware environment and that is the intent of the sections that follow.

4.2.1 Assess Suitability

Assess the suitability of using an integration process for first following the points below:

1. Examine the prerequisites and performance considerations described within the checklist Making Correct Use of Integration Processes (section 4.1.1).

   **Important**

   SAP highly recommends reviewing this checklist (Checklist: Making Correct Use of Integration Processes) in particular as it clearly outlines the correct and intended use for integration processes as well as some clear use cases for when an integration process is not suitable.

2. Review the available example scenarios (section 4.1.2) and, if applicable, apply the example scenario while making appropriate adjustments as necessary.

4.2.2 Not an Answer for Everything

Do not be so quick to use an integration process for moderately complex integration requirements. Listed below are some common integration patterns and scenarios where customers have used integration processes in the past despite the possibility to address the requirements without ccBPM.

**Note**

With an abundance of standard mediation features, SAP NetWeaver PI is equipped to address a variety of integration requirements that, at first glance, might appear as if only an integration process can fulfill. Avoid using integration processes unnecessarily, especially in more technically based, stateless process scenarios (i.e., no correlations).

Splitting Messages

Using multi-mappings for message splits (1:n transformation), n number of resulting messages can be distributed to n number of receiving interfaces and receivers. For more information, check the following resources

- SAP online help: Developing Multi-Mappings for Message Splits
- SDN Blog: Multi-Mapping without BPM

**Note**

Available from XI 3.0 (SP14), PI 7.0 (SP05), PI 7.1 (all SPs)
Collecting Messages

Especially for IDoc scenarios, the sender application can collect IDocs into a single IDoc package and send the IDoc package in one rRFC call to the Integration Server. Alternatively, the sender application could collect the messages into one file using a XML Port and send the file as a single message via the sender File adapter channel (also see Note 814393 “Multiple IDoc instances in an XI message”).

**Recommendation**
Evaluate the overall feasibility of collecting messages on the sender application instead of via ccBPM in general (not just for IDoc scenarios).

**Note**
Starting with PI 7.0 EhP1 and PI 7.1 EhP1, the sender IDoc adapter can process IDoc packages without internally splitting them into one message per individual IDoc contained in the IDoc package.

More Information:
SAP Help Portal: Configuring the Sender IDoc Adapter

Async-Sync and Sync-Async Bridges

For relatively simple scenarios where an async-sync or sync/async bridge is required, the JMS adapter along with a series of corresponding adapter modules can be used.

More Information:
- SAP Help Portal: Configuring Async/Sync and Sync/Async Bridge in the JMS Adapter

Message Enrichment

Messages can be enriched with data from external applications by adding lookups to mapping programs. The PI mapping runtime has a lookup API that supports access to applications using the JDBC, SOAP, or RFC adapters. In addition, starting with PI 7.1, the graphical mapping editor has standard functions for JDBC and RFC lookups which allow for the definition of lookups graphically, thus eliminating the need for any coding when creating a mapping lookup as in previous releases.

More Information:
- SAP Help Portal: Adding Lookups to Mapping Programs

Multiple Sequential Transformations

Multiple sequential transformations can be executed by using an operation mapping with multiple mapping steps. Individual mapping programs can be inserted into an operation mapping. At runtime, the mapping programs are executed as steps in the sequence specified (from top to bottom). The result from the previous step is forwarded to the mapping program of the subsequent step. Although any intermediate payload is not persisted, the mappings are executed using one JCo call (except when an ABAP based mapping program is included).

More Information:
- SAP Help Portal: Preconfiguring Mapping Programs with Operation Mappings
Transformations “Inside” an Integration Process

By executing transformations “outside” the integration process (i.e., prior to entering or prior to leaving an integration process), the performance of an integration process can be significantly improved. Keep in mind that this includes transformation steps that might be used not only at the start or end of the integration process but also those that might be used prior to any Send step, after the Send step (e.g. sync Send step), or after a correlated Receive step. Granted, such measure would not typically replace the need for creating an integration process itself, but would certainly reduce the resource consumption that would otherwise be used.

4.2.3 Take Advantage of New BPE Runtime Features

Leverage improvements in BPE (Business Process Engine – BPE) runtime features to maximize performance. ccBPM runtime governs the handover and processing of PI messages according to an integration process definition in the ESR. New ccBPM runtime features may increase message throughput significantly and provide a more reliable handover of messages to process instances. However, features have to be used carefully and have to be adapted to process design.

⚠️ Note
For an overview of message processing by the BPE, see the Appendix.

4.2.3.1 Message Packaging

Message packaging in the BPE helps improve performance by delivering multiple messages to BPE process instances in one transaction. This can lead to an increased message throughput.

⚠️ Note
Message packaging in BPE is independent of message packaging in PI (XI), but they can be used together. For more information on message packaging in PI, see Message Packaging.
Main Advantages

- Delivery of multiple messages to process instance in a single transaction
- Raises message throughput (but: latency of single message may increase)
- Reduces persistence effort
- Reduces occupied DB space

Use Cases

- Process types awaiting high number of messages
- Process types dealing with message peaks
- Collect Scenarios
  
  Collect Criteria
  
  - Message number based
  - Based on package size
  - Time based

More Information:

- Note 1033205 – BPE: Message Packaging
- SAP Help Portal: Message Packaging for Integration Processes and Using Message Packaging Correctly

4.2.3.2 Delivery Mode

Delivery mode configuration determines whether messages are handed over to a process instance directly or with intermediate buffering.

<table>
<thead>
<tr>
<th>With Buffering</th>
<th>Without Buffering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intermediate message storage if process instance does not provide an open receive step</td>
<td>Process instance has to provide an open receive step when message is handed over to BPE</td>
</tr>
<tr>
<td>All messages that fit to an active correlation are taken up by a running process instance</td>
<td></td>
</tr>
</tbody>
</table>

Message might be captured and stored but is never processed (long running processes, erroneous processes with active correlations)  
If instance does not provide an open receive step, the process type specific queue (XBQO$PE_WS...) goes to status “SYSFAIL”
**Recommendation**

SAP recommends choosing the setting *without buffering* where this is permitted by the semantics of the process. Selecting this setting can help with the analysis of specific problem situations. It will also improve system performance (especially for collect patterns). See use case examples below of how to adapt yet maintain the semantics of a process when using the *without buffering* setting.

**Use Cases**

**With Buffering**

Interrupted synchronous process scheme by *Receive* steps, *Wait* steps, *Send* steps requesting Acknowledgements, *Control* step triggering alerts or steps having set the flag "Create New Transaction".

If the sending application cannot guarantee the sequence of XI messages, a deadlock situation can be induced if *with buffering* option is not used.

**Without Buffering**

Guarantee synchronous process scheme (either by moving the transformation step out of the loop or by deactivating the flag "Create New Transaction".)
If the sending application cannot guarantee the sequence of XI messages, the corresponding receive steps have to be arranged in a fork.

More Information:
- How To Guide: How to Configure Inbound Processing in ccBPM - Part I: Delivery Mode
- SAP Help Portal: Configuring the Delivery Mode for Inbound Processing

### 4.2.3.3 Queue Assignment

Defines whether messages are handed over to process instance using single or multiple queues.

<table>
<thead>
<tr>
<th>Single Queue</th>
<th>Multiple Queues</th>
</tr>
</thead>
<tbody>
<tr>
<td>All messages dedicated to a specific process type are handled by a single queue.</td>
<td>Using multiple queues per process type</td>
</tr>
<tr>
<td>Works for all process definitions</td>
<td>Might conflict with correlation handling</td>
</tr>
</tbody>
</table>

**Settings Overview**
- One Queue (Classic Setting)
- One Configurable Queue

**Settings Overview**
- Multiple Queues (Random)
- Multiple Queues (Content-Specific)
Use Cases (Single Queue)

<table>
<thead>
<tr>
<th>One Queue (Classic Setting)</th>
<th>One Configurable Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Works for all process definitions</td>
<td>Works for all process definitions</td>
</tr>
<tr>
<td>Name of the process type specific queue is changed to XBPE_&lt;Task&gt;</td>
<td>Queue can then be assigned to a dedicated server</td>
</tr>
</tbody>
</table>

**Note**

Setting “One Queue” or “One Configurable Queue” is essential for all processes with more than one correlation.

Use Cases (Multiple Queues)

<table>
<thead>
<tr>
<th>Multiple Queues (Random)</th>
<th>Multiple Queues (Content-Specific)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallelization of process type specific queues</td>
<td>Distribution to multiple queues based on correlations</td>
</tr>
<tr>
<td>Might be used to speed up time critical processes without correlations</td>
<td></td>
</tr>
</tbody>
</table>

**Requirements**

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process must not contain any correlation</td>
<td>Process must contain only one correlation</td>
</tr>
<tr>
<td>The content of that correlation must not change during the life time of a process instance.</td>
<td></td>
</tr>
<tr>
<td>The correlation should be selective (i.e., it should instantiate multiple integration processes in parallel)</td>
<td></td>
</tr>
</tbody>
</table>

More Information:

- How To Guide:  [How to Configure Inbound Processing in ccBPM Part II: Queue Assignment](#)
- SAP Help Portal:  [Configuring Queue Assignment for Inbound Processing](#)

**Transaction Handling**

It is possible to influence transactional behavior in the definition of an integration process in the process editor. This can potentially improve performance. A transaction is a sequence of processing steps that are either fully executed, or not executed at all. If the system cannot fully execute a transaction, it rolls back all processing steps within the transaction. This means that the system can return to the state before the transaction was begun.
CAUTION

The transaction control mechanism for integration processes is not a central transaction manager for an entire system landscape. The transaction control mechanism cannot roll back any processing steps outside the Integration Server.

Following step types support adjustable transaction handling

- Send (without Acknowledgements)
- Receiver determination
- Transformation
- Block

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increased processing performance due to</td>
<td>• Increased burden on main memory for expansive transactions</td>
</tr>
<tr>
<td>• Reduced number of work items</td>
<td>• Less monitoring information available</td>
</tr>
<tr>
<td>• Reduced number of DB transactions</td>
<td>• Restart at the last sync point</td>
</tr>
</tbody>
</table>

More Information:

- How To Guide: [How to Configure ccBPM Runtime Part III: Transactional Behavior of an Integration Process](#)
- [Note 1082099](#): BPE-TS: Transactional behavior
- SAP Help Portal: [Transactional Behavior of an Integration Process](#)
5. Additional Design Topics

5.1 Usage of Local Software Component Version

You can define a local software component version freely in the Enterprise Services Repository (see Creating a Local Software Component Version).

Local software component versions are intended for temporary development of local objects in an ES Repository only.

Local objects cannot be used in a configuration in the Integration Directory, and subsequently cannot perform any runtime tests with these objects.

No local objects can be transported using the Change Management Service (CMS).

**Recommendation**

Local software component versions are useful for conducting proof of concept. Users can experiment on various modeling techniques in the local SWCV objects. Once the proof of concept is required in the real business scenario, all objects from LSCV can be transferred to the SLD software component via release transfer and used further.

**More Information:**
- SAP Help Portal: Software Component Versions

5.2 When to Use Outside-In and Inside-Out Approach for Defining Services

**Inside-Out**

In order to reuse the already existing implementation or functionality such as RFC-enabled function modules, BAPIs, or IDOCs, we can use the inside-out approach. This will prevent to invent the wheel again.

**More Information:**
- SAP Help Portal: Developing a Web Service in Inside-Out Mode
- SAP Help Portal: Developing Enhanced Services Inside-Out

**Outside-In**

SAP recommends the outside-in approach for any new development.

Outside-In approach helps to define the business processes much in advance. We can model the business scenario in PI, identify the process, and define services, interfaces and data types to be used. Here all objects reside in one central repository belonging to a business scenario. Once services are defined, they get implemented at the backend. Modeling is independent of the backend. Hence user has a choice to implement the services in any backend as per their requirement.

**More Information:**
- SAP Help Portal: Generating Interfaces and Proxies – The Outside-In Approach
6. Appendix

Appendix A – Overview – Message Processing by BPE

The following illustration and corresponding steps describe how messages are handled and processed by the Business Process Engine (BPE).

1. A message is handed over to the process type specific queue (XBQO$PE_WS...).
2. Queue gathers the messages and handles them sequentially.
3. Correlation handling detects that a new process instance has to be created.
4. The process instance is executed according to the process type definition.
5. The process type specific queue still receives messages for that process type. No handover of messages until the synchronous processing of the active process instance is interrupted.
6. Receive or wait step releases the queue for processing of the intercepted messages.
7. Following message of queue is handled.

Steps 1 to 7 are executed sequentially for each message being sent to this process type.
8. Next message starts a new process instance (example). Process Instance 1 waits for the following message.

9. The last message is transferred to this instance and is processed following the process definition until its end. Process Instance 1 terminates and the queue takes over the control again.

Appendix B – Queue Assignment (Multiple Queues) Examples

The following examples illustrate use cases for applying the BPE inbound processing queue assignment setting for multiple queues.

### Multiple Queues (Random)

- Async-sync bridges / Lookup scenarios
- Sync-async bridges without response messages
- Sync-sync bridges / Sync-multisync bridges
- Multicasting / Split scenarios without response messages

### Multiple Queues (Content-Specific)

- Collect scenarios
- Single request-response scenarios