

SAP NetWeaver Process Integration 7.1 Mapping Enhancements



SAP NetWeaver Regional Implmentation Group
SAP NetWeaver Product Management
December 2007

THE BEST-RUN BUSINESSES RUN SAP™





After reading the document, you will be able to:

- **Know the new features of the graphical mapping tool in PI 7.1**
- **Work with variables**
- **Parameterize the mappings during configuration**
- **Use the function libraries for User Defined Functions**
- **Create database lookups with help of the graphical support**

Agenda



1. **Miscellaneous Features**
2. **Graphical Variables**
3. **Parameterized Mappings**
4. **Development of User-Defined Functions**
5. **Functional Libraries**
6. **Graphical Support for Lookups**



Feature	Details	Example
Interface Mappings are now called Operation Mappings		
Choice between SAP XML toolkit and jdk 1.5 toolkit	<ul style="list-style-type: none"> ■ For compatibility reasons SAP XML toolkit is still available ■ Only relevant for Java and XSLT mappings 	
Set failure behavior in functions FixValues and Value mapping	Functions may: <ul style="list-style-type: none"> ■ Return the initial value ■ Return a default ■ Throw an exception 	
Output of fields and functions can be used for multiple target fields	<ul style="list-style-type: none"> ■ reuse of complex combinations of functions ■ Simplify refactoring of mappings with repeating combinations of functions ■ Better runtime performance 	

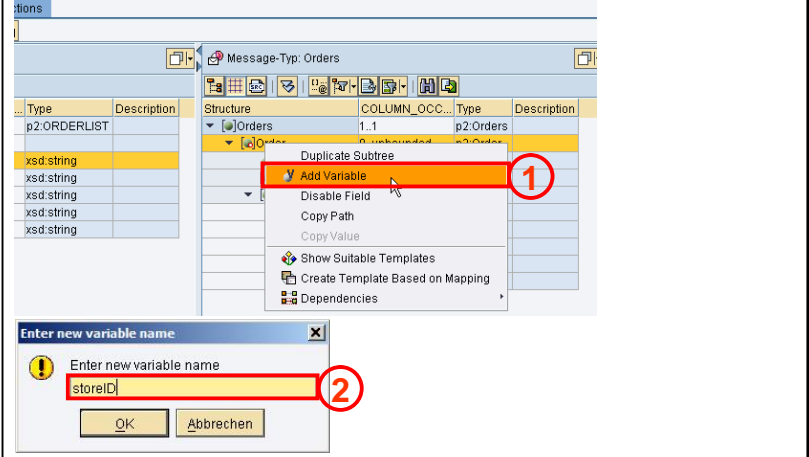
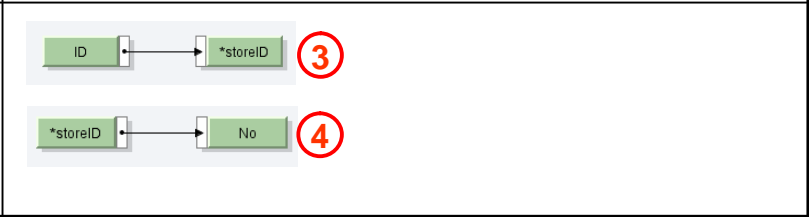


Feature	Details	Example																																				
Refactoring capabilities of Message Mappings	<ul style="list-style-type: none"> ■ Tool support to adjust mappings after structure changes (e.g. changes of the root node name, changes of the namespace in Message Types) ■ Applies to changes of the source and target structure ■ „Map“ original node to new node 	<p>The screenshot shows the SAP Message Mapping tool interface. At the top, there are tabs for 'Definition', 'Test', 'Signature', and 'Functions'. Below the tabs is a toolbar with various icons. The main area displays 'Message-Typ: METAORDERLIST'. A table shows the structure of the message type:</p> <table border="1"> <thead> <tr> <th>Structure</th> <th>COLUMN_OCC...</th> <th>Type</th> <th>Des</th> </tr> </thead> <tbody> <tr> <td>[-] METAORDERLIST</td> <td>1..1</td> <td>p2:METAORDERLIST</td> <td></td> </tr> <tr> <td>[-] LIST</td> <td>1..1</td> <td>p2:ORDERLIST</td> <td></td> </tr> <tr> <td>[-] Item</td> <td>0..unbounded</td> <td></td> <td></td> </tr> <tr> <td>[-] ID</td> <td>1..1</td> <td>xsd:string</td> <td></td> </tr> <tr> <td>[-] POSITION</td> <td>1..1</td> <td>xsd:string</td> <td></td> </tr> <tr> <td>[-] PRODUCT</td> <td>1..1</td> <td>xsd:string</td> <td></td> </tr> <tr> <td>[-] QUANTITY</td> <td>1..1</td> <td>xsd:string</td> <td></td> </tr> <tr> <td>[-] PRICE</td> <td>1..1</td> <td>xsd:string</td> <td></td> </tr> </tbody> </table> <p>Below the table is a 'Move Field' dialog box. It has a 'Missing Field' section with two radio buttons: <input type="radio"/> /ns0:ORDERLIST/Item/ID and <input checked="" type="radio"/> /ns0:ORDERLIST/Item/. Below this is a tree view of the structure, with 'Item' selected. At the bottom, there are 'Source' and 'Target' fields and buttons for 'Move', 'Skip', and 'Skip All'.</p>	Structure	COLUMN_OCC...	Type	Des	[-] METAORDERLIST	1..1	p2:METAORDERLIST		[-] LIST	1..1	p2:ORDERLIST		[-] Item	0..unbounded			[-] ID	1..1	xsd:string		[-] POSITION	1..1	xsd:string		[-] PRODUCT	1..1	xsd:string		[-] QUANTITY	1..1	xsd:string		[-] PRICE	1..1	xsd:string	
Structure	COLUMN_OCC...	Type	Des																																			
[-] METAORDERLIST	1..1	p2:METAORDERLIST																																				
[-] LIST	1..1	p2:ORDERLIST																																				
[-] Item	0..unbounded																																					
[-] ID	1..1	xsd:string																																				
[-] POSITION	1..1	xsd:string																																				
[-] PRODUCT	1..1	xsd:string																																				
[-] QUANTITY	1..1	xsd:string																																				
[-] PRICE	1..1	xsd:string																																				

Graphical Variables



- Store intermediate mapping results (including context information)
- Reuse in multiple target field mappings
- Enhanced runtime performance

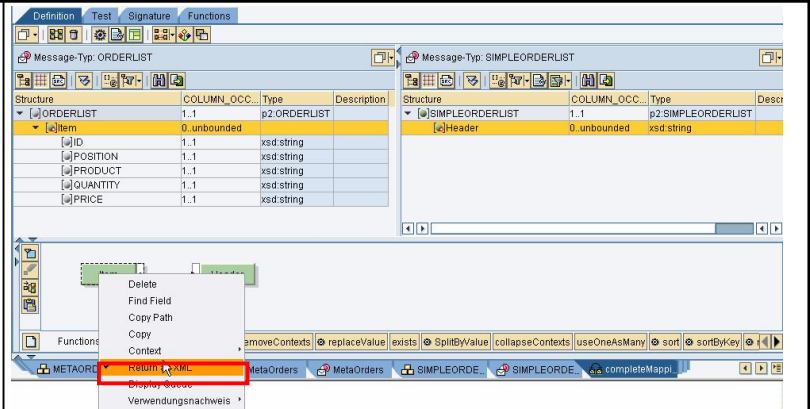
<ol style="list-style-type: none">1. Right-click on element of your target structure and select Add Variable2. Enter a name for the variable	
<ol style="list-style-type: none">3. Define a mapping to set the intermediate variable4. The variable can be used in all subsequent mappings. It will not be written to the final document	

Copy entire XML Subtree

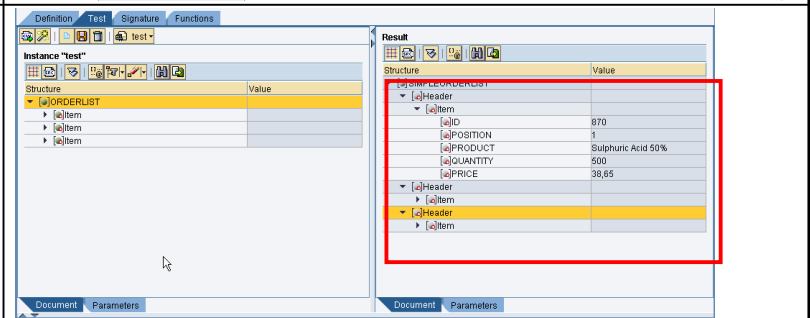


- Complete copy of XML subtrees
- Target structure not mandatory
- Occurrence of the target node has to correspond with the source node

1. Set up a target field mapping that links the relevant nodes
2. Right-click on the node of your source structure and select **Return as XML**




3. The entire node (including the underlying XML branch) will be copied to the target node. The target node will take over the occurrence of the source node.

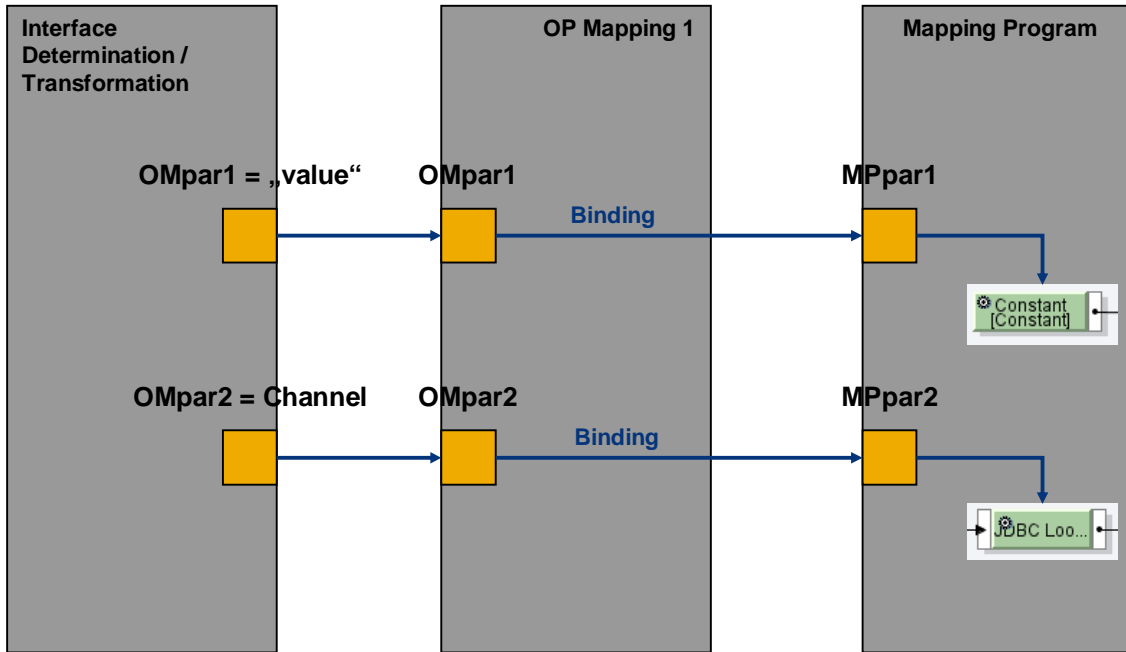


Parameterized Mappings



- Function Properties and Parameters in User-defined Functions (denoted by the  icon) can be set in Interface Determinations and Transformation Steps of Integration Processes and Monitoring Processes (**import** parameters)
- Payload information can be written to simple typed containers of an Integration Process and Monitoring Processes (**export** parameters)
- Provide **simple typed constants** or **channels** (for lookup functions)
- Parameters can be used in
 - Message Mappings
 - Java Mappings
 - XSLT Mappings
- Examples:
 - Reuse of mappings in multiple Interface Determinations with varying parameterizations
 - Set Channel for lookups at configuration time
 - Transfer content of simple typed containers to the message payload

Parameterized Mappings – Binding



Parameterized Mappings in Message Mappings - Doing



1. Add list of parameters in the Signature tab of your Message Mapping
2. Set a Name, Category, Type and the direction for the relevant parameter
3. Double click on the function which provides a property / parameter
4. Select the parameter defined in step 2

The screenshot shows the SAP Message Mapping tool interface. The 'Signature' tab is selected, showing 'Source Message(s)' and 'Target Message(s)'. Below these are two tables for message details. The 'Parameters' table is highlighted with a red box and contains the following data:

Name	Category	Type	Parameter	Description
MMadapter1	Adapter	JDBC	Import	
MMpar1	Simple Type	xsd:string	Import	

A 'Constant' Parameters dialog box is open, showing a 'Value' field with '\$MMpar1\$'. The 'MMpar1' parameter is selected in the list below. Red circles 1, 2, 3, and 4 highlight the 'Signature' tab, the 'Parameters' table, the 'Constant' dialog, and the selected parameter respectively.

Parameterized Mappings in Operation Mappings – Doing



1. Select "Parameters" in your **Operation Mapping**
2. Set a Name, Category, Type and the direction for the relevant parameter
3. Select „Binding“ to set the dependencies between Message Mapping and Operation Mapping parameters

Name	Category	Type	Parameter	Description
OMadapter1	Adapter	JDBC	Import	
OMpar1	Simple Type	xs:string	Import	

Type	Name	Namespace	Binding	Orders
Mapping Program	firstMapping	http://sap.com/DK	Import	Orders

4. Set the binding

Message Mapping Parameter	Binding
MMadapter1	OMadapter1
MMpar1	OMpar1

Parameterized Mappings in Interface Determination – Doing



1. Create your **Interface Determination** and add the relevant Operations Mapping
2. Assign constants or channels to your parameter

The screenshot shows the 'Edit Interface Determination' window in SAP. The 'Sender' section is configured with 'Kommunikationspartner' (empty), 'Kommunikationskomponente' (DKT_Sender), 'Interface' (ORDERLIST), and 'Namensraum' (http://sap.com/DKT/mappingtest). The 'Receiver' section is configured with 'Kommunikationspartner' (empty), 'Kommunikationskomponente' (DKT_Receiver), and 'Description' (empty). The 'Software Component Version of Sender Interface' is set to 'J0_SOFTWARECOMP 1.0 of jo.com'. The 'Maintain Order At Runtime' checkbox is checked. The 'Receiver Interfaces' table is shown below:

Condition	Operation Mapping	Name *	Namespace *	Multiplicity
	firstOperationMapping	Orders	http://sap.com/DKT/mapp	1

Below the table, the 'Parameter of Operation-Mapping: firstOperationMapping | http://sap.com/DKT/mappingtest' is shown:

Name	Value
OMadapter1	JDBCAdapter JDBCReceiverSync
OMpar1	'myValue'

Red circles and boxes highlight the 'firstOperationMapping' entry in the table (labeled '1') and the parameter table (labeled '2').

Parameterized Mappings in Integration Process – Doing



1. Create a **Transformation Step** in your Integration Process or Monitoring Process.
2. Assign constant values or simple typed containers to import parameters
3. Assign simple typed containers to export parameters

Properties	
Name	Value
Step Name	Transformation
Description	
Operation Mapping	absOperationMapping
Create New Transaction	<input type="checkbox"/>
▼ Exceptions	
System Error	
▼ Source Messages	
ORDERLIST_abs http://sap.	ORDERLIST
▼ Target Messages	
Orders_abs http://sap.com/!	Orders
▼ Import	
OMPar1_in	'myValue'
▼ Export	
OMPar1_out	SimpleContainer

2

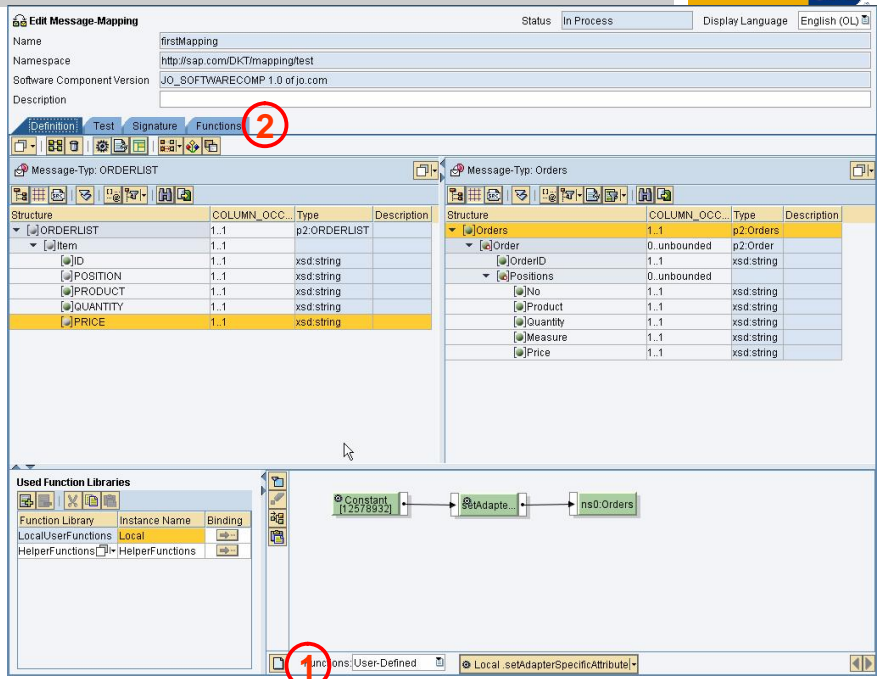
3

New Development of User-Defined Functions (UDF)



To create or change a UDF, either:

1. Select „Create New Function“
- or
2. Select the „Functions“ tab



UDF Development - Using "Create New Function"



1. Set a (technical) **Name** and a (functional) **Title** of your UDF
2. Set the **Execution Type** for the required context handling
3. Specify a **Category**
4. Define **Arguments** (input variables) and **Parameters** (function properties). For Parameters you may set a **Title** in addition
5. Select „**Create Function**“
6. Add **import** statements (if necessary)
7. Develop your coding

The image shows two screenshots from the SAP NetWeaver IDE. The top screenshot is the 'New Function' dialog box. It has fields for Name (setAdapterSpecificAttribute), Title (setAdapterSpecificAttribute), Description, Execution Type (Single Values), and Category (AdapterAttributeFuncs). Below these is a 'Signature Variables' table:

Type	Name	Java Type	Title
Argument	attValue	String	
Parameter	attName	String	Attribute Name
Parameter	attNamespace	String	Attribute Namespace

The bottom screenshot is the 'Show Function' editor for 'setAdapterSpecificAttribute'. It shows the following code:

```
import
java.text.ParseException
java.util.Date
java.util.Calendar

public String setAdapterSpecificAttribute(String attValue, String attName, String attNamespac
    DynamicConfiguration conf = (DynamicConfiguration) container.getInputHeader().getA
    DynamicConfigurationKey key = DynamicConfigurationKey.create(attNamespace, attN
    conf.put(key, attValue);

    return "";
}
```

UDF Development - Using the "Functions" tab



1. Specify a (technical) **Name**
2. Specify a (functional) **Title**, the **Execution Type** for context handling and a **Category**
3. Define **Arguments** (input variables) and **Parameters** (function properties). For Parameters you may set a **Title** in addition
4. Adapt the **import** statement (if necessary)
5. Develop your coding

The screenshot shows the SAP UDF development interface for the class `_firstMapping_` in package `com.sap.xi.tf`. The **Functions** tab is active, displaying the following details:

- Functions and Java Areas:** A list of functions including `init`, `cleanUp`, `calcWeek`, and `setAdapterSpecificAttribute` (highlighted with a red circle 1).
- Import Instructions:** A list of imported classes including `com.sap.aii.mapping.api.*`, `java.io.*`, `java.lang.reflect.*`, `java.util.*`, `java.text.DateFormat`, `java.text.ParseException`, `java.util.Date` (highlighted with a red circle 4), and `java.util.Calendar`.
- Signature Variables:** A table defining the function's parameters:

Type	Name	Java Type	Title
Argument	attValue	String	
Parameter	attName	String	Attribute Name
Parameter	attNamespace	String	Attribute Namespace

(The `attName` parameter is highlighted with a red circle 3.)
- Function Properties:** The `setAdapterSpecificAttribute` function is configured with:
 - Title:** `setAdapterSpecificAttribute` (highlighted with a red circle 2)
 - Execution Type:** `Single Values`
 - Category:** `AdapterAttributeFuncs`
- Code Editor:** The function implementation is shown as:

```
public String setAdapterSpecificAttribute(String attValue, String attName, String attNamespace, Container
DynamicConfiguration conf = (DynamicConfiguration) container.getInputHeader().getAtAll();
DynamicConfigurationKey key = DynamicConfigurationKey.create(attNamespace, attName);
conf.put(key, attValue);
```

(The code editor is highlighted with a red circle 5.)



Use-cases

- Reuse of user-defined functions
- Enhanced portability of user-defined functions

To set up a Library:

1. Provide **Function Library** name
2. Set **Import Instruction** / add reference to **Imported Archive**
3. Add a new (technical) **function** name and provide (functional) **Title** (may differ)
4. Specify a function **Category**
5. Set signature variables (**Arguments** and **Parameters**)
6. Develop Java code
7. Use **Import/Export** buttons to transfer Java source code from/to Java IDE (e.g. the NWDS)

The screenshot shows the 'Edit Funktionsbibliothek' (Edit Function Library) configuration screen. The interface is annotated with red circles and numbers 1 through 7, corresponding to the steps in the list above.

- 1**: Name field containing 'HelperFunctions'.
- 2**: 'Import Instructions' list containing 'com.sap.aui.mapping.api.*', 'com.sap.aui.mapping.lookup.*', 'com.sap.aui.mappingtools.rtt.*', 'java.io.*', 'java.lang.reflect.*', 'java.util.*', 'com.sap.guid.*', and 'java.math.BigInteger'.
- 3**: 'Name' field containing 'flexibleSubstring'.
- 4**: 'Category' dropdown menu set to 'TextExtension'.
- 5**: 'Signature Variables' table with columns 'Type', 'Name', 'Java Type', and 'Title'. It contains three rows: 'startPos' (Argument, String), 'endPos' (Argument, String), and 'var' (Argument, String).
- 6**: Java code editor showing the implementation of the 'flexibleSubstring' function.
- 7**: 'Import/Export' buttons.

Type	Name	Java Type	Title
Argument	startPos	String	
Argument	endPos	String	
Argument	var	String	

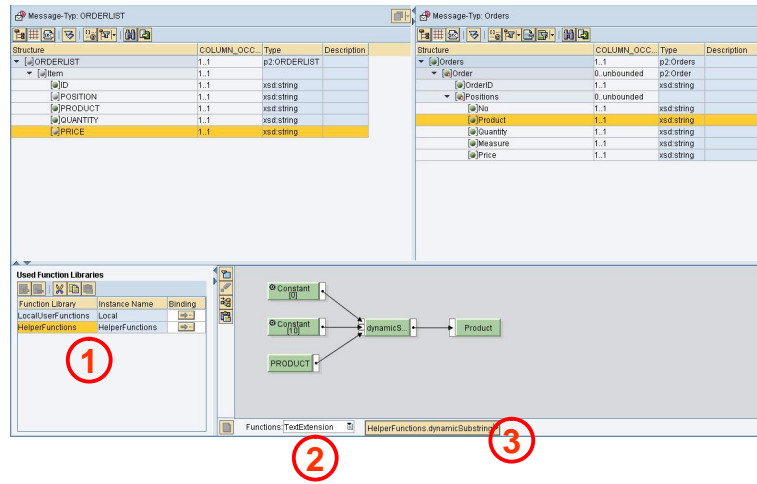
```

public String flexibleSubstring(String startPos, String endPos, String var, Container container) throws StreamTransformationExc
{
    Integer startPosInt = new Integer(startPos);
    Integer endPosInt = new Integer(endPos);
    var = var.trim();
    if (var.length() > 0) {
        return var.substring(startPosInt.intValue(), var.length());
    } else {
        return var.substring(startPosInt.intValue(), endPosInt.intValue());
    }
}
    
```

Use a Function Library



1. Add the relevant **Function Library** to the library list. You may select all libraries of the actual and underlying SCVs
2. The **Categories** of the Function Libraries show up in the drop down menu
3. Select the relevant **Category** and the relevant **Function**



Function Library Entities



Display Funktionsbibliothek

Name: **HelperFunctions**

Software Component Version: **JO_SOFWARECOM1.0 of jo.com**

Class Name: **Library** | Package Name: **com.sap.dikt.mapping.test**

Function Name: **DynamicSubString**

Category: **ExtExtension**

Signature Variables:

Type	Name	Java Type	Title
Argument	startPos	String	
Argument	endPos	String	
Argument	var	String	

```

public String flexibleSubString(String startPos, String endPos, String var, Container container) throws StreamTransformationException {
    Integer startPosInt = new Integer(startPos);
    Integer endPosInt = new Integer(endPos);
    var = var.trim();
    if (startPosInt <= endPosInt.intValue()) {
        return var.substring(startPosInt.intValue(), endPosInt.intValue());
    } else {
        return var.substring(endPosInt.intValue(), endPosInt.intValue());
    }
}
    
```

Function Library

Entities

- Class Name
 - Package Name
 - Function Name
- are technical (Java) constructs

Message Typ: ORDERLIST

Structure	COLUMN_OCC.	Type	Description
ORDERLIST	1..1	xs:ORDERLIST	
Item	1..1	xs:ITEM	
Item	1..1	xs:ITEM	
Item	1..1	xs:ITEM	
Item	1..1	xs:ITEM	
Item	1..1	xs:ITEM	

Message Typ: Orders

Structure	Type	Description
Orders	xs:ORDER	
Order	xs:ORDER	
Product	xs:PRODUCT	
Quantity	xs:QUANTITY	
Price	xs:PRICE	

Function Library Name: **HelperFunctions**

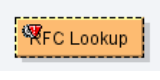
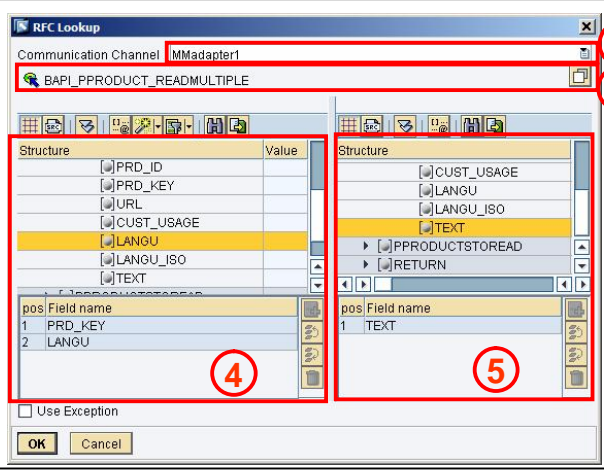
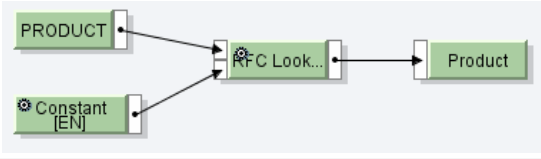
Function Title: **DynamicSubString**

Category: **ExtExtension**

Message Mapping

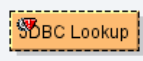
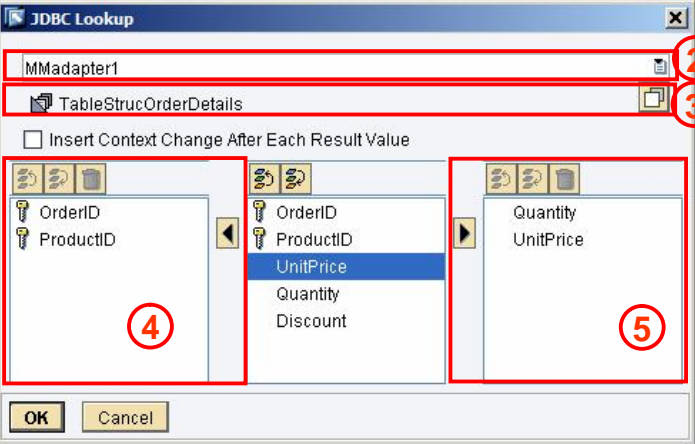
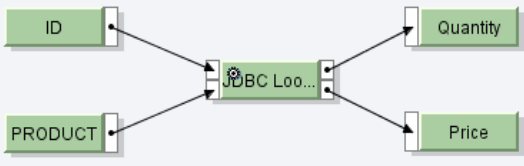
Graphical Support for RFC Lookups



<ol style="list-style-type: none"> 1. Select function RFC Lookup from category Conversions. Double click on function. 	
<ol style="list-style-type: none"> 2. Select a Communication Channel from your parameter list (see Parameterized Mappings for details) 3. Select the relevant RFC structure (has to be imported beforehand) 4. Specify request parameters for the RFC 5. Select the required response parameters of the RFC 	
<ol style="list-style-type: none"> 6. Finish your target field mapping 	

Graphical Support for JDBC Lookups



<ol style="list-style-type: none"> 1. Select function JDBC Lookup from category Conversions. Double click on function. 	
<ol style="list-style-type: none"> 2. Select a Communication Channel from your parameter list 3. Select the relevant table structure (has to be defined as External Definition beforehand) 4. Specify select parameters for the SQL statement 5. Specify the resultset of your SQL statement 	
<ol style="list-style-type: none"> 6. Finish your target field mapping 	

Summary



- You have learned the new features of the graphical mapping tool in PI 7.1
- You can work with variables
- You can parameterize the mappings during configuration
- You can use the function libraries for User Defined Functions
- You can create database lookups with help of the graphical support



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, Excel, 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, System i, System i5, System p, System p5, System x, System z, System z9, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWERS, 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.

The information in this document is proprietary to SAP. No part of this document may be reproduced, copied, or transmitted in any form or for any purpose without the express prior written permission of SAP AG.

This document is a preliminary version and not subject to your license agreement or any other agreement with SAP. This document contains only intended strategies, developments, and functionalities of the SAP® product and is not intended to be binding upon SAP to any particular course of business, product strategy, and/or development. Please note that this document is subject to change and may be changed by SAP at any time without notice.

SAP assumes no responsibility for errors or omissions in this document. SAP does not warrant the accuracy or completeness of the information, text, graphics, links, or other items contained within this material. This document is provided without a warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement.

SAP shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials. This limitation shall not apply in cases of intent or gross negligence.

The statutory liability for personal injury and defective products is not affected. SAP has no control over the information that you may access through the use of hot links contained in these materials and does not endorse your use of third-party Web pages nor provide any warranty whatsoever relating to third-party Web pages.