

How to Lookup Data Via a RFC User Defined Function?



Applies to:

To use the mapping lookup API you need SAP NetWeaver™ Exchange Infrastructure with at least SP13 or higher.

Summary

This paper provides details on the usage of the Generic Lookup API used for calling RFC from user defined Java functions in SAP PI Integration Builder.

As of SAP NetWeaver '04 SP13 a new mapping lookup API is available which simplifies using RFC lookups.

This guide assumes you already have some basic SAP PI knowledge.

Author: Danny De Roovere

Company: SAP Belgium Luxembourg

Created on: 12 March 2008

Author Bio



Danny De Roovere is a principal SAP NetWeaver consultant employed at SAP Belgium-Luxembourg. Danny started as a SAP basis consultant (in particular EDI-ALE), and immediately added a new dimension to his career when SAP released its NetWeaver platform. Today, his main focus is SAP's business-driven software architecture (eSOA), SAP Enterprise Portal, SAP Exchange Infrastructure and Duet.

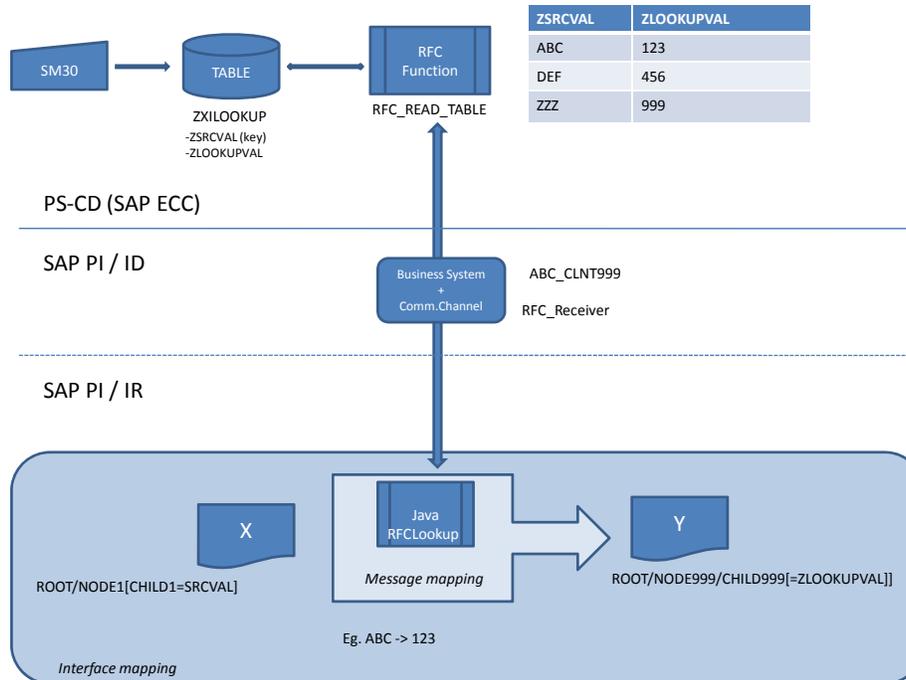
Table of Contents

Prerequisites.....	3
Scenario Overview.....	3
Backend Configuration data	4
Table ZXILOOKUP (se11)	4
Table entry (sm30).....	4
Function Module RFC_READ_TABLE (se37)	5
PI Development / Configuration.....	6
Function Module.....	6
User Defined Function	8
RFC accessor and XML parser.....	9
Message Mapping.....	11
Related Content.....	12
Copyright.....	13

Prerequisites

To use the mapping lookup API you need SAP NetWeaver™ Exchange Infrastructure with at least SP13 or higher.

Scenario Overview



The scenario demonstrates the look-up process.

Data in one of the child elements of an XML document (X) is taken as input for one of the child elements of XML document (Y). The data is not passed as such; the goal is to retrieve its equivalent data value from a qualified lookup table.

The table is created and maintained in the ABAP Business System. In our scenario this table only contains two columns: ZSRCVAL (input parameter) and ZLOOKUPVAL (output parameter).

The standard RFC Function Module, RFC_READ_TABLE, is responsible for the handling of the lookup request. The RFC Call is executed in a Java User Defined function using the RFC communication channel. This enables us not only to monitor the RFC adapter in case of any errors (using the Runtime Workbench) but also to store the passwords inside the channel and not inside the code.

Backend Configuration data

Table ZXILOOKUP (se11)

The following table will act as a lookup table for all XI value mappings

Dictionary: Maintain Table

Transp. Table: ZXILOOKUP Inactive(Revised)
 Short Description: SAP XI Lookup table

Attributes | Delivery and Maintenance | **Fields** | Entry help/check | Currency/Quantity Fields

Field	Key	Initi...	Data element	Data Ty...	Length	Decimal Places
ZSRCVAL	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	ZSRCVAL	CHAR	35	0
ZLOOKUPVAL	<input type="checkbox"/>	<input type="checkbox"/>	ZLOOKUPVAL	CHAR	35	0
	<input type="checkbox"/>	<input type="checkbox"/>				

It consists of:

- ZSRCVAL: source value (key)
- ZLOOKUPVAL: value to be returned

Note that the user that will access the table in read-mode (in our case an RFC user) needs to have the authorization profile S_TABU_DIS.

Table entry (sm30)

Via transaction SM30 the table can be maintained (add rows, delete rows, modify rows).

Function Module RFC_READ_TABLE (se37)

The table entries will be fetched from ECC via the standard function module RF_READ_TABLE. With transaction se37 you can test the function module.

Test Function Module: Initial Screen

Debugging Test data directory

Test for function group SDTX
 Function module RFC_READ_TABLE
 Uppercase/Lowercase

RFC target sys:

Import parameters	Value
QUERY_TABLE	ZXILOOKUP
DELIMITER	
NO_DATA	
ROWSKIPS	0
ROWCOUNT	0

Tables	Value
OPTIONS	<input type="checkbox"/> 0 Entries
FIELDS	<input type="checkbox"/> 0 Entries
DATA	<input type="checkbox"/> 0 Entries

We'll pass three parameters to the function module:

- QUERY_TABLE: the name of the table. In our case **ZXILOOKUP**
- OPTIONS > TEXT: the 'WHERE' statement from the SQL query. E.g. **ZSRCVAL = 'ABC'** (this can be extended with eg the AND operand)
- FIELDS: the column to be returned: In our case **ZLOOKUPVAL**.

PI Development / Configuration

Function Module

At first you have to import the RFC function module RFC_READ_TABLE because you need its signature in the java User Defined function:

You are on the design maintenance screen in the Integration Builder.

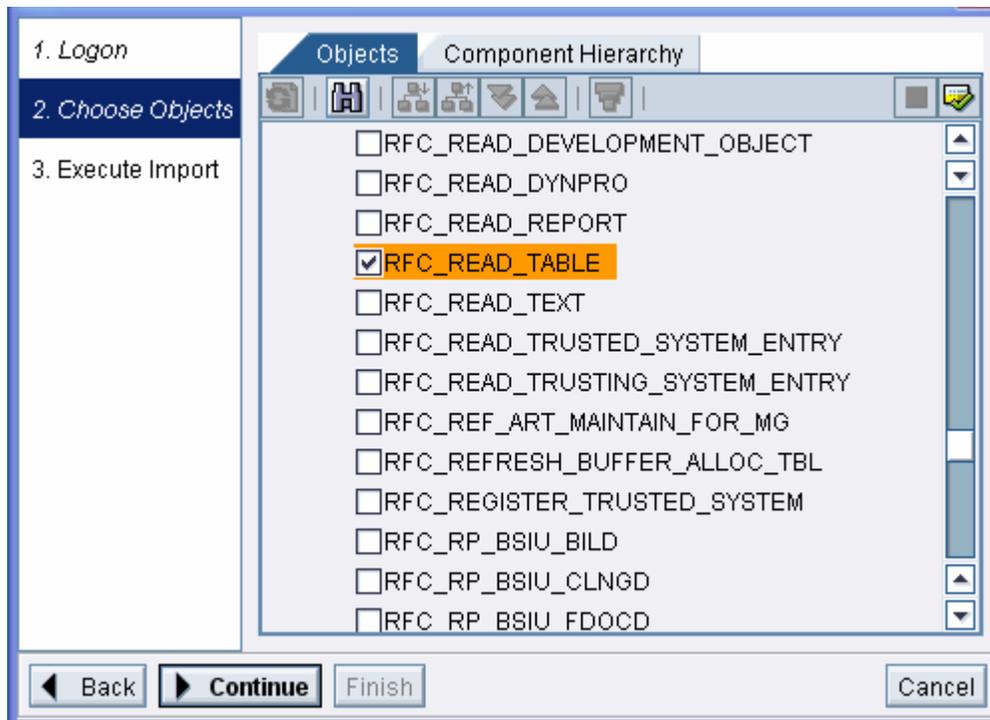
Expand the subnodes for the software component version for which you want to import interfaces.

Call the context menu for the *Imported Objects* subnode of this software component version and choose *Import of SAP Objects*.

The first step of the import wizard is *Logon*. Enter the user that you want to use to import the interfaces from the SAP system.

The second step of the import wizard is *Choose Objects*. Select the object RFC_READ_TABLE to import. Note that it takes approximately one minute before all objects are displayed. The import wizard displays the objects on the *Objects* tab page (sorted alphabetically) and on the *Component Hierarchy* tab page (sorted by application components.)

In the third step, *Execute Import*, you choose *Continue* and *Finish*.



Once the function imported, you need its RFC XML signature.

A trick to lookup the signature is to create a 1-1 Message Mapping for the function module and run a test.

Map a constant value in the element 'TEXT'. The constant represents the WHERE_CLAUSE of the SQL statement to be carried out.

The screenshot shows the SAP ABAP IDE interface. The top part displays the structure of the RFC message 'RFC_READ_TABLE' in two views: a tree view on the left and a table view on the right. The table view has columns for Occurrences, Type, and Details. The 'TEXT' element is highlighted in orange in both views.

Below the structure view, a mapping diagram shows a constant value 'ZSRCVAL = ABC' being mapped to the 'TEXT' element. A 'Constant Properties' dialog box is open, showing the value 'ZSRCVAL = ABC' and the notation 'Any string value'.

The bottom toolbar contains various functions: Text, substring, concat, equalsS, indexOf, indexOf, lastIndexOf, lastIndexOf, compare, replaceString, length, endsWith, startsWith, startsWith, toUpperCase, trim, toLowerCase.

To lookup the XML signature click, test the mapping and click on the source icon in the result pane.

The screenshot shows the SAP ABAP IDE interface. The top part displays the structure of the RFC message 'RFC_READ_TABLE' in two views: a tree view on the left and a table view on the right. The 'TEXT' element is highlighted in orange in both views.

Below the structure view, a mapping diagram shows a constant value 'ZSRCVAL = ABC' being mapped to the 'TEXT' element. A 'Constant Properties' dialog box is open, showing the value 'ZSRCVAL = ABC' and the notation 'Any string value'.

The bottom toolbar contains various functions: Text, substring, concat, equalsS, indexOf, indexOf, lastIndexOf, lastIndexOf, compare, replaceString, length, endsWith, startsWith, startsWith, toUpperCase, trim, toLowerCase.

The bottom part of the screenshot shows the 'Result' pane, which displays the XML signature of the RFC call. The XML string is:

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:RFC_READ_TABLE xmlns:ns0="urn:sap-com:document:sap:rfc:functions"><QUERY_TABLE>ZXILOOKUP</QUERY_TABLE><DATA></DATA><FIELDS><item><FIELDNAME></FIELDNAME></item></FIELDS><OPTIONS><item><TEXT>ZSRCVAL = &apos;ABC&apos;</TEXT></item></OPTIONS></ns0:RFC_READ_TABLE>
```

Cut and paste the XML string in e.g. Notepad. You need it later.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:RFC_READ_TABLE xmlns:ns0="urn:sap-com:document:sap:rfc:functions"><QUERY_TABLE>ZXILOOKUP</QUERY_TABLE><DATA></DATA><FIELDS><item><FIELDNAME></FIELDNAME></item></FIELDS><OPTIONS><item><TEXT>ZSRCVAL = &apos;ABC&apos;</TEXT></item></OPTIONS></ns0:RFC_READ_TABLE>
```

User Defined Function

As we'll be using the RFC lookup from a user defined function we have to create it first (for each message mapping that will make use of this function).

Our function will be a simple type function with 5 inbound arguments. To create a new user-defined function, in the data-flow editor, choose *Create New Function*, which is located on the lower left-hand side of the screen. In the menu, choose *Simple Function* or *Advanced Function*.

In the window that appears, specify the attributes of the new function:

- *Label*
Technical name of the function. The name is displayed in the function chooser and on the data-flow object.
- *Description*
Description of how the function is used.
- *Cache*
Function type
- *Arguments*
In this table, you specify the number of input values the function can process, and name them. All functions are of type *String*.

Our function will be a simple type function with 5 arguments:

Type	Name	Java type
Argument	DBTABLE	String
Argument	ZSRCVAL	String
Argument	FIELDS	String
Argument	businessSystem	String
Argument	communicationChannel	String

DBTABLE	The name of the lookup table
ZSRCVAL	The value to be looked up in the table
FIELDS	Columns to be returned by the RFC function module (RFC_READ_TABLE)
businessSystem	The application server that hosts the RFC function module
communicationChannel	The RFC receiver communication channel used for the connection between SAP PI and the backend.

RFC accessor and XML parser

In order to use the API you have to add the following packages to the User defined function:

- RFC accessor
 - com.sap.aii.mapping.lookup.*
 - java.io.*
- XML parser
 - javax.xml.parsers.DocumentBuilder
 - org.xml.sax.*
 - javax.xml.transform.dom.DOMSource
 - javax.xml.parsers.*
 - org.w3c.dom.*
 - javax.xml.transform.stream.StreamResult
 - javax.xml.transform.*

```
// Danny De Roovere
// SAP Belgium
// March 12, 2008

// declare parameters
String returnValue = "";
String WHERE_CLAUSE = "ZSRCVAL = &apos;" + ZSRCVAL + "&apos;";
String rfcXML = "<?xml version='1.0' encoding='UTF-8'?'><ns0:RFC_READ_TABLE
xmlns:ns0='urn:sap-com:document:sap:rfcfunctions'><QUERY_TABLE>" + DBTABLE +
"</QUERY_TABLE><DATA></DATA><FIELDS><item><FIELDNAME>" + FIELDS +
"</FIELDNAME></item></FIELDS><OPTIONS><item><TEXT>" + WHERE_CLAUSE +
"</TEXT></item></OPTIONS></ns0:RFC_READ_TABLE>";

AbstractTrace trace = container.getTrace();
RfcAccessor accessor = null;
ByteArrayOutputStream out = null;

try
{

// 1. Determine a communication channel (Business system + Communication channel)
Channel channel = LookupService.getChannel(businessSystem,communicationChannel);

// 2. Get a RFC accessor for the channel.
accessor = LookupService.getRfcAccessor(channel);

// 3. Create an XML input stream that represents the RFC request message.
InputStream inputStream = new ByteArrayInputStream(rfcXML.getBytes());

// 4. Create the XML Payload
XmlPayload payload = LookupService.getXmlPayload(inputStream);

// 5. Execute the lookup.
Payload result = null;
result = accessor.call(payload);
InputStream in = result.getContent();

// 6. Create a DOM structure from the input XML
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
```

Replace it by your own
WHERE_CLAUSE for your
lookup

RFC XML signature.
Cut-and-paste from
Notepad and replace the
string in the TEXT element
by the WHERE_CLAUSE
parameter

```
Document document = builder.parse(in);
NodeList list = document.getElementsByTagName("WA"); // The lookupValue is available
as WA tag in the response
Node node = list.item(0);
if (node != null) {
    node = node.getFirstChild();
    if (node != null) {
        returnValue = node.getNodeValue();
    }
}

// 7. To free resources, close the accessor..
if (accessor!=null) {
    try {
        accessor.close();
    } catch (LookupException e) {
        trace.addWarning("Error while closing accessor " + e.getMessage() );
    }
}

} catch (Exception e) {
    trace.addWarning("Error" + e);
}

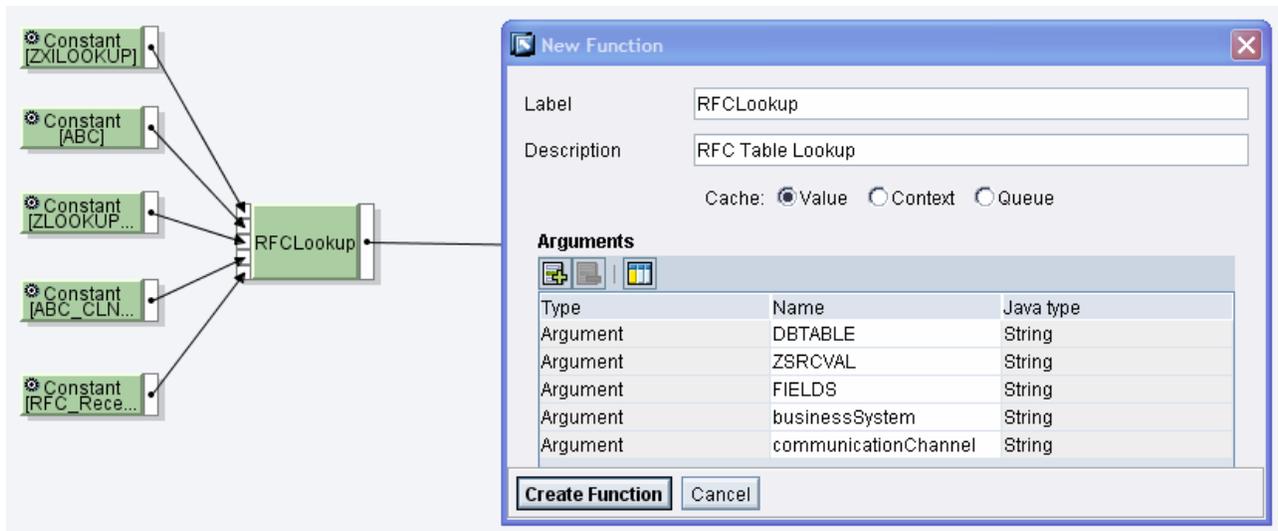
// 8. return a single id value to the message mapping
return returnValue;
```

Message Mapping

During message mapping the source value will be passed to the user defined function parameter **ZSRCVAL**. The returned result is the corresponding value that is maintained in the table ZXILOOKUP in ECC (**ZLOOKUPVAL**).

The other parameters to be passed are constant values:

- **DBTABLE**: name of the table. In our case ZXILOOKUP
- **FIELDS**: column to be returned. In our example we are only interested in one column hence ZLOOKUPVAL
- **businessSystem**: business system where the RFC function module is hosted (to be setup in the Integration Directory)
- **communicationChannel**: RFC receiver communication channel that contains all connection parameters (to be setup in the Integration Directory)



To be complete, an interface mapping needs to be created as well (and used in the interface determination during runtime). Also the scenario

Useful URL's

Using JDBC Connection Pool in XI
Message Mapping:

<https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/6183>

RFC_READ_TABLE :

SAP Note Number: 382318

User Defined Functions :

http://help.sap.com/saphelp_nw04/helpdata/en/22/e127f28b572243b4324879c6bf05a0/frameset.htm

Related Content

Sample Code and API documentation: <https://help.sap.com/javadocs/pi/SP3/xpi/index.html>

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, 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, POWER5, POWER5+, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

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

Oracle is a registered trademark of Oracle Corporation.

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

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

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

Java is a registered trademark of Sun Microsystems, Inc.

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

MaxDB is a trademark of MySQL AB, Sweden.

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

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

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

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

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

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