SDN Community Contribution

(This is not an official SAP document.)

Mapping lookups – RFC API

Applies To:

SAP Netweaver™ Exchange Infrastructure SP13

Summary

This article provides details on the usage of Generic Lookup API used for calling RFC from user defined functions.

Special thanks for Maciej Klimkowski and Marcin Galczynski who help me a lot with Java issues in the Exchange Infrastructure.

By: Michal Krawczyk

Company: BCC

Date: 16 Aug 2005

Table of Contents

Introduction

As of SAP NetWeaver '04 SP13 a new mapping lookup API is available which simplifies using RFC, JDBC and SOAP lookups. It can be executed inside: graphical message mappings (user defined functions) , Java and XSLT mappings. In this article I'll try to show how RFC lookups from user defined functions can be created.

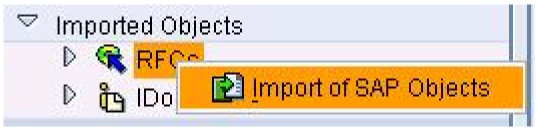
Prerequisites

To use mapping lookup API you need SAP Netweaver™ Exchange Infrastructure with at least SP13.

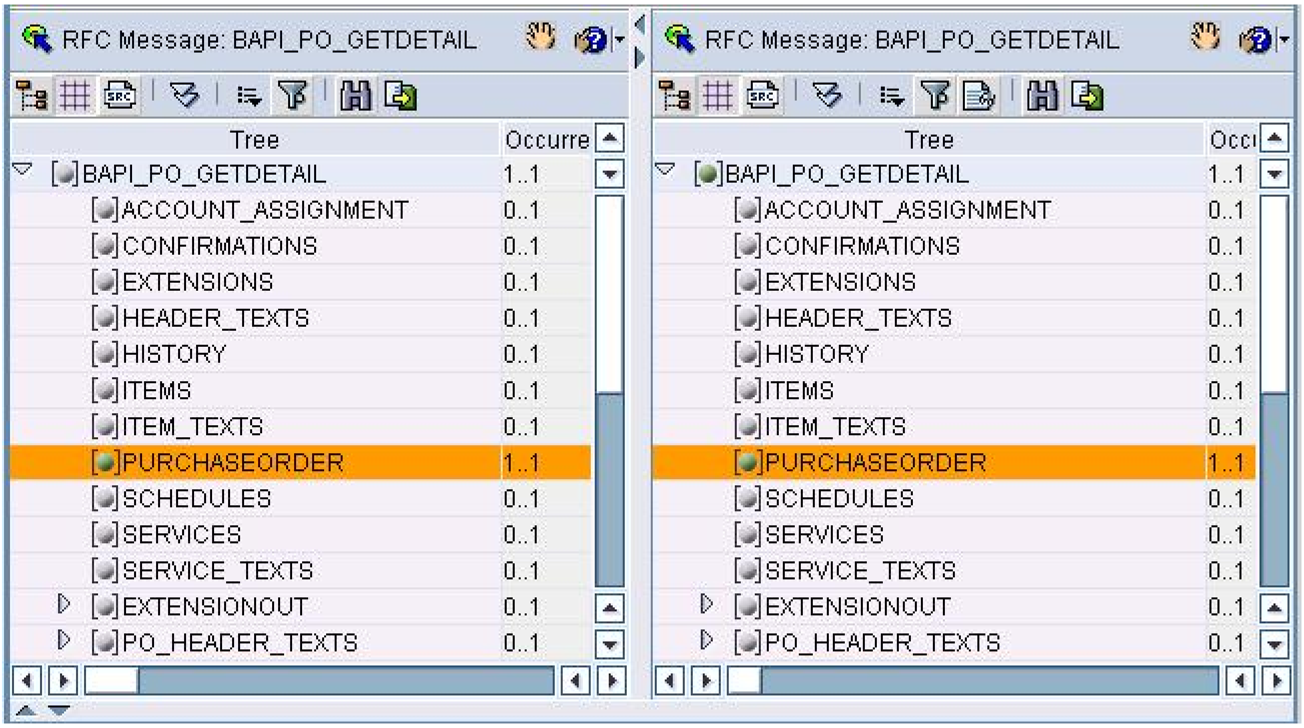
Importing RFC signature

RFC mapping lookup is a “Generic Lookup API” that’s why at first we need to build the RFC payload that the adapter expects. You can do it with three steps.

At first import your BAPI, or RFC into Integration Repository.



Then create a message mapping in which both of your messages are the same RFC message and map the values that you need when calling the RFC in R/3.

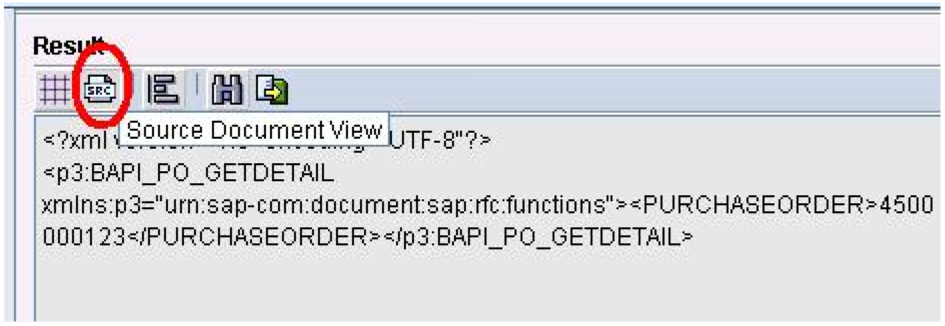

RFC Message: BAPI_PO_GETDETAIL

Tree	Occurre
BAPI_PO_GETDETAIL	1..1
ACCOUNT_ASSIGNMENT	0..1
CONFIRMATIONS	0..1
EXTENSIONS	0..1
HEADER_TEXTS	0..1
HISTORY	0..1
ITEMS	0..1
ITEM_TEXTS	0..1
PURCHASEORDER	1..1
SCHEDULES	0..1
SERVICES	0..1
SERVICE_TEXTS	0..1
EXTENSIONOUT	0..1
PO_HEADER_TEXTS	0..1

RFC Message: BAPI_PO_GETDETAIL

Tree	Occı
BAPI_PO_GETDETAIL	1..1
ACCOUNT_ASSIGNMENT	0..1
CONFIRMATIONS	0..1
EXTENSIONS	0..1
HEADER_TEXTS	0..1
HISTORY	0..1
ITEMS	0..1
ITEM_TEXTS	0..1
PURCHASEORDER	1..1
SCHEDULES	0..1
SERVICES	0..1
SERVICE_TEXTS	0..1
EXTENSIONOUT	0..1
PO_HEADER_TEXTS	0..1

Then test your message mapping in the Test tab filling the values that you need. When you change the Result view to the “Source Document View” and you can copy the XML-RFC message that we will use in our mapping lookup.


Result

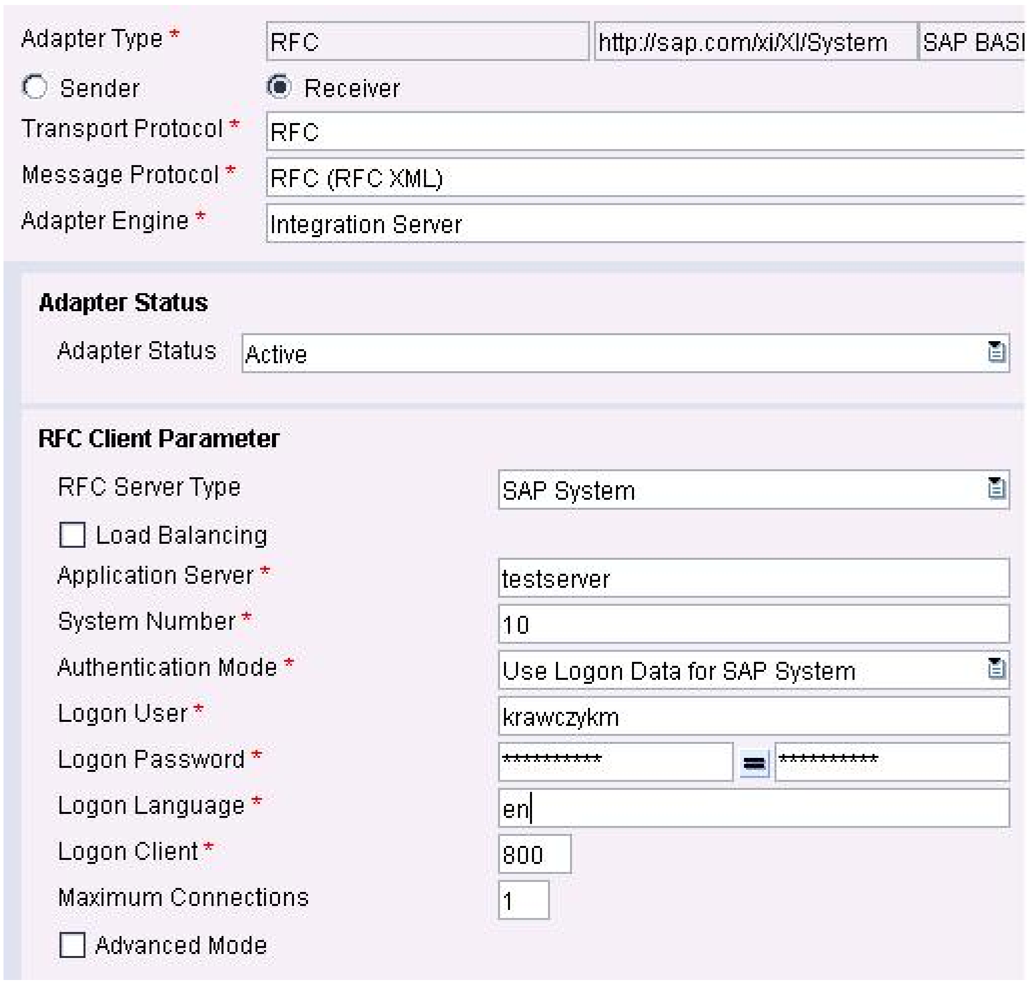
Source Document View

```
<?xml v                UTF-8"?>
<p3:BAPI_PO_GETDETAIL
xmlns:p3="urn:sap-com:document:sap:rfc:functions"><PURCHASEORDER>4500
000123</PURCHASEORDER></p3:BAPI_PO_GETDETAIL>
```

Communication Channel

During our RFC call we'll be using 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 as it was in the past when you wanted to use SAP data lookups within XI mappings.

RFC receiver communication channel configuration:



Adapter Type *	RFC	http://sap.com/xi/XI/System	SAP BASI
Sender	[✓ Receiver]		
Transport Protocol *	RFC		
Message Protocol *	RFC (RFC XML)		
Adapter Engine *	Integration Server		

Adapter Status

Adapter Status	Active

RFC Client Parameter

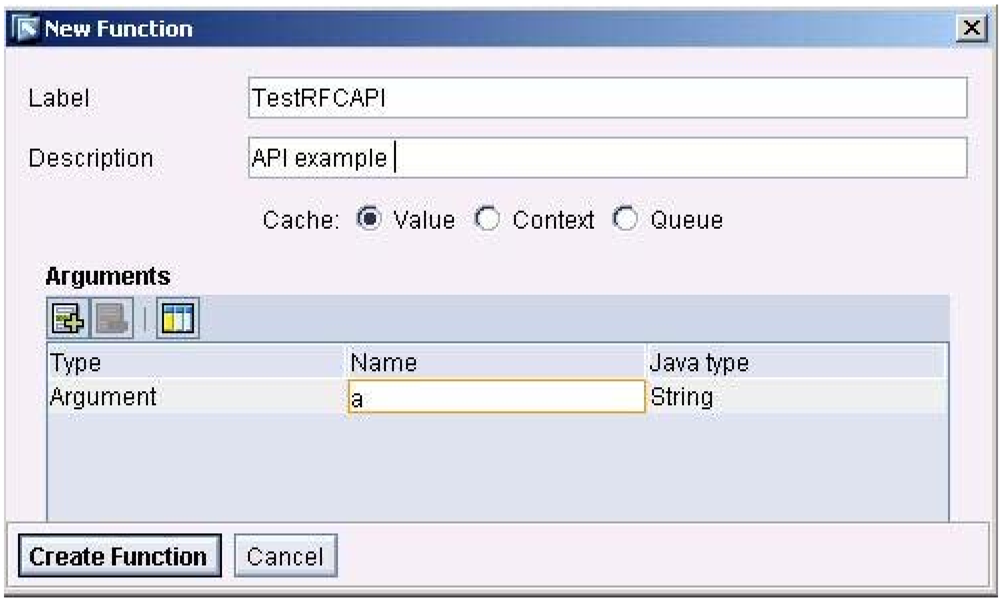
RFC Server Type	SAP System
☐ Load Balancing	
Application Server *	testserver
System Number *	10
Authentication Mode *	Use Logon Data for SAP System
Logon User *	krawczykm
Logon Password *	*********** = ***********
Logon Language *	en
Logon Client *	800
Maximum Connections	1
☐ Advanced Mode	

Mapping lookup - RFC accessor

As we'll be using RFC lookup from user defined function we have to create it first.

Our function will be a simple type function with one inbound argument. We won't be using the argument as we have our RFC payload with values. This article does not cover building the RFC call inside the function nor parsing the result. We'll just call the RFC and return the XML-RFC result.

Creating Simple user defined function:


New Function

Label: TestRFCAPI

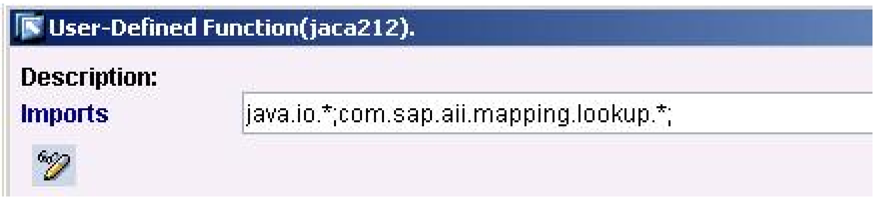
Description: API example

Cache: [✓ Value] Context Queue

Arguments

Type	Name	Java type
Argument	a	String

In order to use our example API you have to add: java.io.* and com.sap.aii.mapping.lookup.* packages to our User defined function.


User-Defined Function(jaca212).

Description:

Imports: java.io.*;com.sap.aii.mapping.lookup.*;

I decided to use a standard BAPI (BAPI_PO_GETDETAIL) so that everyone who has connection to R/3 can try this example without to many modifications.

This is the a code sample:

```
String content = "";

MappingTrace importanttrace;

importanttrace = container.getTrace();

// filling the string with our RFC-XML (with values)

String m = "<?xml version=\"1.0\" encoding=\"UTF-8\"?><p2:BAPI_PO_GETDETAIL
xmlns:p2=\"urn:sap-
com:document:sap:rfc:functions\"><PURCHASEORDER>4500014790</PURCHASEORDER></p2:B
API_PO_GETDETAIL>";

RfcAccessor accessor = null;

ByteArrayOutputStream out = null;

try

      {

            // 1. Determine a channel (Business system, Communication channel)

            Channel channel = LookupService.getChannel("BCS_800","rfc_channel");

            // 2. Get a RFC accessor for a channel.

            accessor = LookupService.getRfcAccessor(channel);

            // 3. Create a xml input stream representing the function module
request message.

            InputStream inputStream = new ByteArrayInputStream(m.getBytes());

            // 4. Create xml payload

            XmlPayload payload = LookupService.getXmlPayload(inputStream);

            // 5. Execute lookup.

            Payload result = accessor.call(payload);

            InputStream in = result.getContent();

            out = new ByteArrayOutputStream(1024);

            byte[] buffer = new byte[1024];

            for (int read = in.read(buffer); read > 0; read = in.read(buffer)) {

                  out.write(buffer, 0, read);

            }

            content = out.toString();
      }
catch(LookupException e)
      {
            importanttrace.addWarning("Error while lookup " + e.getMessage() );
      }
catch(IOException e)
      {
            importanttrace.addWarning("Error " + e.getMessage() );
      }
finally
      {
            if (out!=null) {
                  try {
                         out.close();
                  } catch (IOException e) {
                        importanttrace.addWarning("Error while closing stream "
+ e.getMessage() );
                  }
            }
            // 7. close the accessor in order to free resources.
            if (accessor!=null) {
                  try {
                         accessor.close();
                  } catch (LookupException e) {
                        importanttrace.addWarning("Error while closing accessor
" + e.getMessage() );
                  }
            }
      }
```

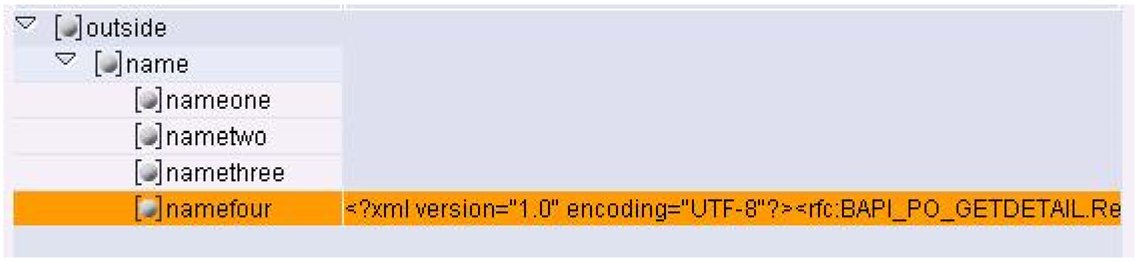
```
//returning the result – RFC-XML.response

return content;
```

You may also find sample code and API documentation on JAVADOC index page.

Results

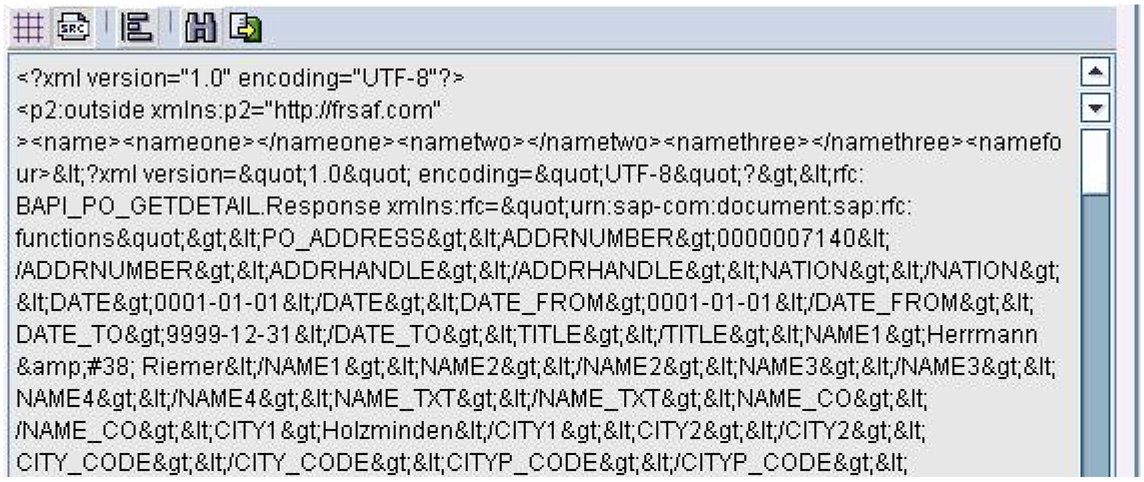
Now we’re ready to test our function.

As I mentioned in this article we only map the RFC.response to one of the XML tags



```
[ ]outside
  [ ]name
    [ ]nameone
    [ ]nametwo
    [ ]namethree
    [ ]namefour   <?xml version="1.0" encoding="UTF-8"?><rfc:BAPI_PO_GETDETAIL.Re
```

If you change to the “Source Document View” you can see the whole response.



```
<?xml version="1.0" encoding="UTF-8"?>
<p2:outside xmlns:p2="http://frsaf.com"
><name><nameone></nameone><nametwo></nametwo><namethree></namethree><namefo
ur>&lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;&lt;rfc:
BAPI_PO_GETDETAIL.Response xmlns:rfc=&quot;urn:sap-com:document:sap:rfc:
functions&quot;&gt;&lt;PO_ADDRESS&gt;&lt;ADDRNUMBER&gt;0000007140&lt;
/ADDRNUMBER&gt;&lt;ADDRHANDLE&gt;&lt;/ADDRHANDLE&gt;&lt;NATION&gt;&lt;/NATION&gt;
&lt;DATE&gt;0001-01-01&lt;/DATE&gt;&lt;DATE_FROM&gt;0001-01-01&lt;/DATE_FROM&gt;&lt;
DATE_TO&gt;9999-12-31&lt;/DATE_TO&gt;&lt;TITLE&gt;&lt;/TITLE&gt;&lt;NAME1&gt;Herrmann
&amp;#38; Riemer&lt;/NAME1&gt;&lt;NAME2&gt;&lt;/NAME2&gt;&lt;NAME3&gt;&lt;/NAME3&gt;&lt;
NAME4&gt;&lt;/NAME4&gt;&lt;NAME_TXT&gt;&lt;/NAME_TXT&gt;&lt;NAME_CO&gt;&lt;
/NAME_CO&gt;&lt;CITY1&gt;Holzminden&lt;/CITY1&gt;&lt;CITY2&gt;&lt;/CITY2&gt;&lt;
CITY_CODE&gt;&lt;/CITY_CODE&gt;&lt;CITYP_CODE&gt;&lt;/CITYP_CODE&gt;&lt;
```

To get the values that you want you only need to parse this XML.

Author Bio

Michal Krawczyk is a certified Development Consultant SAP - Exchange Infrastructure 3.0 Developer and Development Consultant SAP NetWeaver - ABAP Workbench 2003 at BCC. His areas of expertise include Exchange Infrastructure, interfaces, IDoc's, RFC's, ALE, XML. His interests are new integration applications including XI and AII