SDN Community Contribution

(This is not an official SAP document.)

Disclaimer & Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.
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.

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.
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:

<table>
<thead>
<tr>
<th>Adapter Type *</th>
<th>RFC</th>
<th><a href="http://sap.com/xi/XiSystem">http://sap.com/xi/XiSystem</a></th>
<th>SAP BASIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sender</td>
<td>Receiver</td>
<td>RFC</td>
<td></td>
</tr>
<tr>
<td>Transport Protocol *</td>
<td>RFC</td>
<td>RFC (RFC XML)</td>
<td>Integration Server</td>
</tr>
<tr>
<td>Message Protocol *</td>
<td>RFC</td>
<td>RFC (RFC XML)</td>
<td>Integration Server</td>
</tr>
<tr>
<td>Adapter Engine *</td>
<td>RFC</td>
<td>RFC (RFC XML)</td>
<td>Integration Server</td>
</tr>
</tbody>
</table>

- **Adapter Status**: Active

- **RFC Client Parameter**
  - **RFC Server Type**: SAP System
  - **Load Balancing**: testserver
  - **Application Server ***: 10
  - **System Number ***: Use Logon Data for SAP System
  - **Authentication Mode ***: krawczyk
  - **Logon User ***: ************
  - **Logon Password ***: ************
  - **Logon Language ***: en
  - **Logon Client ***: 800
  - **Maximum Connections**: 1
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:

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.

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:BAPI_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

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

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 All