Accessing Adapter-Specific Attributes through User Defined Function

Summary

This article features the procedure of accessing adapter specific attributes provided by some adapters (precisely filename in a file adapter). Presented here is a simple scenario that helps in understanding the basic concept.

Author: Latika Sethi

Company: HCL Technologies

Created on: 5th April, 2007

Author Bio

The author, Latika Sethi, is presently working with HCL Tech. as SAP NetWeaver XI consultant.

Table of Contents

Adapter Specific Message Attributes

In a business scenario a need could arise to access some adapter specific parameters at runtime. There value is either not known until runtime or simply needs to be changed/accessed at runtime. This can be done by mapping runtime constants by using them to get access to the java classes for adapter specific parameters.

In the header of the XI message, there are adapter-specific parameters. These can be used by the sender adapter to assign a value to them. The process is very simple. One needs to get access to the parameters during mapping and assign them the desired values that are available at runtime. The whole code can be just written in a user defined function that is called during the execution of message mapping.

Let's get into a little depth…

In order to get access to an adapter-specific parameter, one needs to know:

- The namespace of the adapter
- The name of the attribute.

The namespace for the adapter is nothing but the namespace in Integration Repository where the adapter metadata is saved followed by the name of the adapter. File adapter, for example, has the following namespace: http://sap.com/xi/XI/System/File.

The package which contains the classes for accessing adapter-specific attributes is com.sap.aii.mapping.api

One requires the following classes:

com.sap.aii.mapping.api.DynamicConfigurationKey (to create a key having adapter namespace and parameter name, which is used in getting/setting the value of the parameter)

com.sap.aii.mapping.api.DynamicConfiguration (contains key-values pairs for the attributes, is used to modify the value of a parameter).

The attributes to be accessed need to be explicitly set during the configuration of the adapter. That is if the sender adapter needs to add some attributes to message header, these must be indicated in sender adapter configuration. Likewise, if receiver adapter needs to read the attributes, these must be set in receiver adapter configuration.

Getting access to FileName in file adapter

There are various adapter specific attributes available for file adapter. For sender adapter: filename, directory, file type, source file size, source file timestamp. For receiver: Filename, directory, file type.

The steps for accessing the parameter “FileName” of file adapter are given here. We give the name of the file while configuring the sender file adapter. If the parameter “FileName”, under adapter specific message attributes, is checked in the communication channel, the source file name will be present in the header of the XI message. This name can then be accessed by a code written in a function and the value can be used according to the requirement.

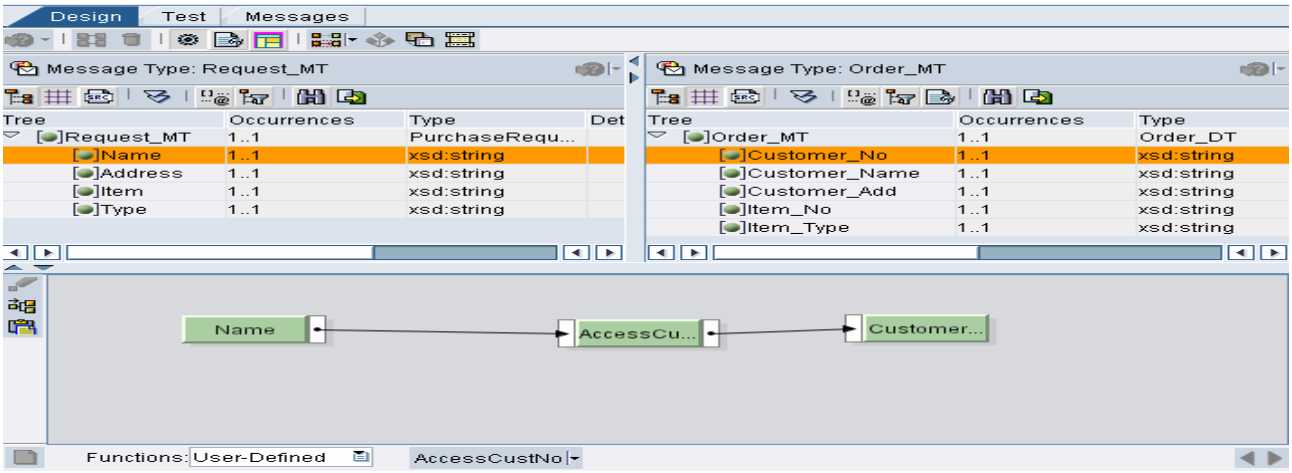
[We can see this parameter in header of the message (Message Content in message Monitoring)]

IR

Let’s have a look at a very simple example where an order request is mapped to the order data type. Suppose the order file from the customers has a number as its name and the customer number in database is the combination of customer name and this number. Hence, the name of the input file is to be extracted during the execution of the mapping program.

Create the source and target data types, message types and the message interfaces.

Create the message mapping as shown:



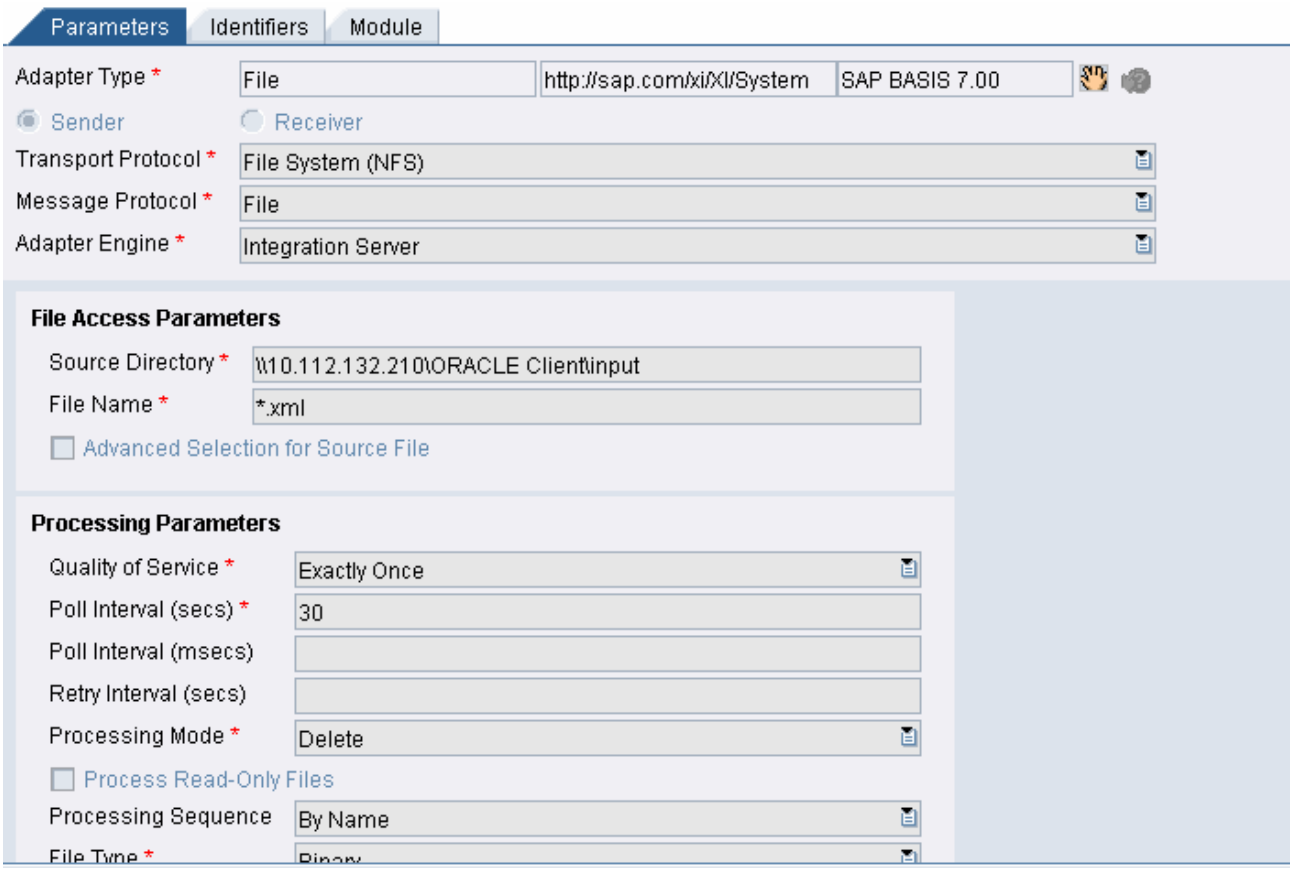
AccessCustNo is the user defined function which extracts the file name at runtime. The code is given in section Sample Code.

Customer_No in target message is Name + FileName (refer to the code)

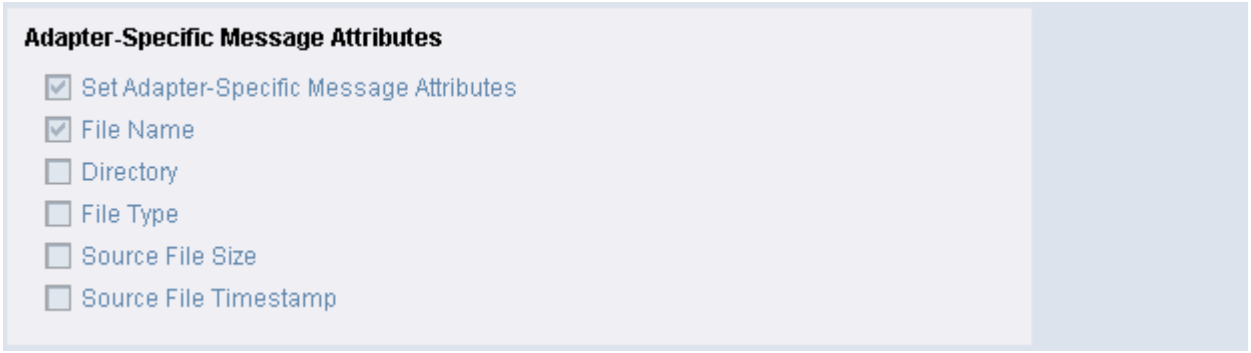
ID

Configure the sender and receiver file adapter as shown below:

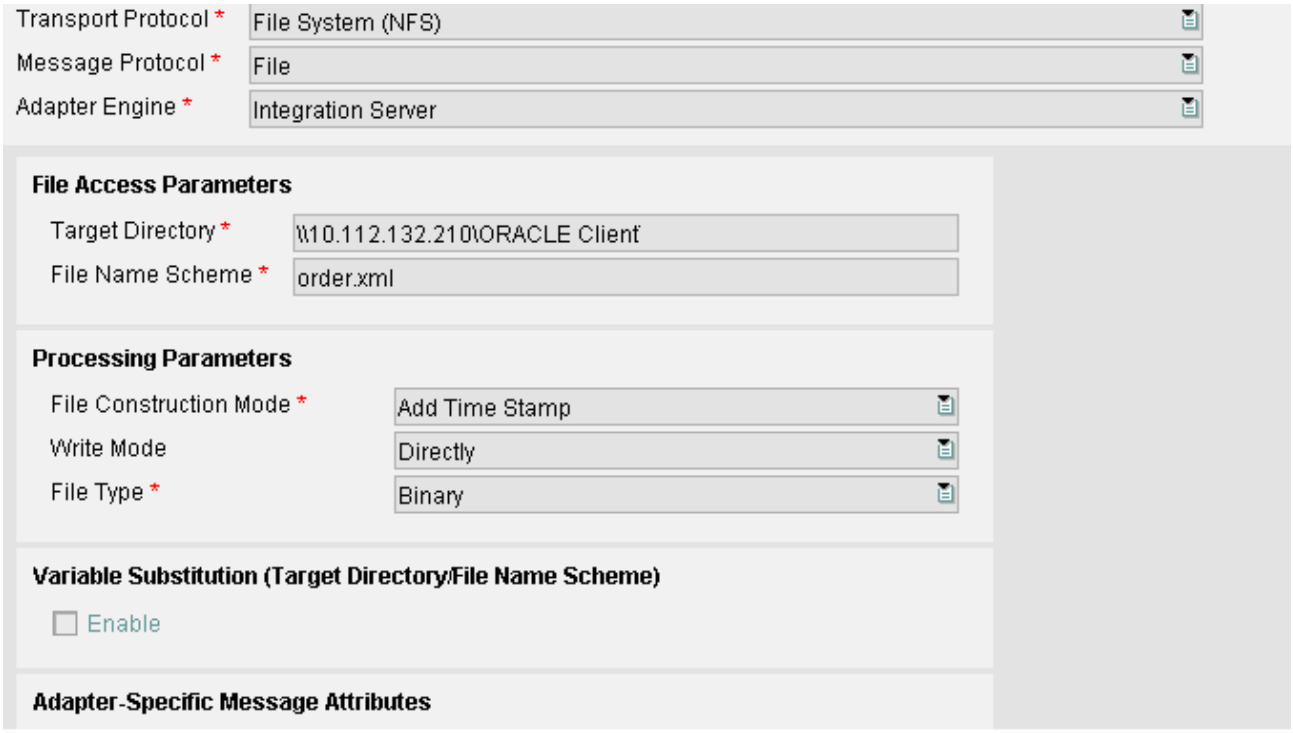
Sender configuration: transport protocol can be either NFS or FTP depending on whether the file is posted on integration server or on some other location.



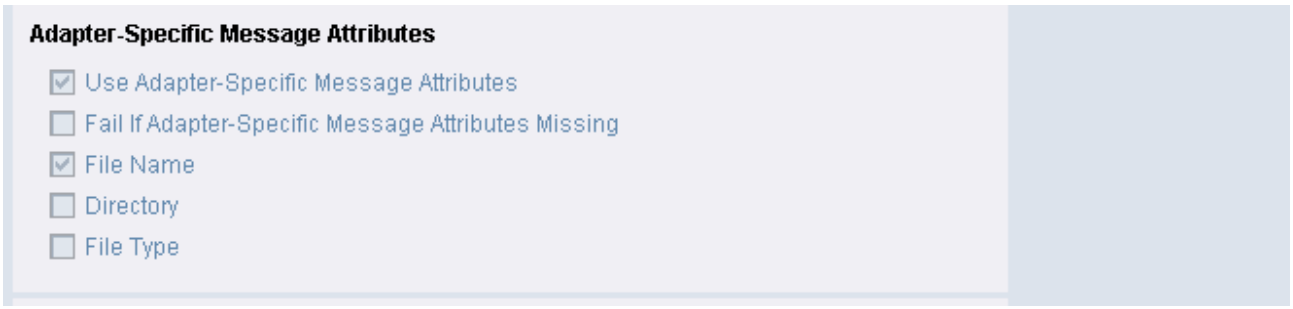
Set the adapter specific parameters



Configuration of the receiver



Set adapter specific parameters



This makes the filename (modified) available in the message header which will be the name of the output file.

Create the Receiver Determination, Interface Determination, Sender agreement, Receiver agreement.

Testing the Scenario

Place the input file in source directory: the name of the file in this case is 100126 (the customer's number)

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ns0:Request_MT xmlns:ns0="urn:legacy:filetofile">
    <Name>MAX</Name>
    <Address>Dell City</Address>
    <Item>Printer</Item>
    <Type>DD</Type>
  </ns0:Request_MT>
```

The file should be deleted after processing as the processing mode in sender adapter is Delete.

The output file created will be:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ns0:Order_MT xmlns:ns0="urn:legacy:filetofile">
    <Customer_No>MAX100126</Customer_No>
    <Customer_Name>MAX</Customer_Name>
    <Customer_Add>Dell City</Customer_Add>
    <Item_No>Printer</Item_No>
    <Item_Type>DD</Item_Type>
  </ns0:Order_MT>
```

Note that the Customer_No is Name + Input file name

Sample Code

Imports: com.sap.aii.mapping.api.DynamicConfigurationKey;com.sap.aii.mapping.api.DynamicConfiguration;

```
public void AccessCustNo(String[] Name,ResultList result,Container container){
  //write your code here
String CustName = Name[0];
 DynamicConfiguration attrib =
(DynamicConfiguration)container.getTransformationParameters().get(StreamTransformationConstants.DYNAMIC_CONFIGURATION);
// to create a key
DynamicConfigurationKey fileKey = DynamicConfigurationKey.create("http://"+"/sap.com/xi/XI/System/File","FileName");

String CustNo = attrib.get(fileKey);
if (CustNo != null) {
   String Cust = CustName +CustNo;
Cust = Cust.substring(0,Cust.indexOf('.'));

   attrib.put(fileKey, Cust);
result.addValue(Cust);
}
```


Related Content

For related information, visit

https://help.sap.com/javadocs/NW04/current/pi/index.html

http://help.sap.com/saphelp_erp2005vp/helpdata/en/43/0a7d1be4e622f3e10000000a1553f7/content.htm

http://help.sap.com/saphelp_erp2005vp/helpdata/en/43/09b16006526e72e10000000a422035/content.htm

http://help.sap.com/saphelp_erp2005vp/helpdata/en/43/03fe1bdc7821ade10000000a1553f6/content.htm

Disclaimer and 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.