Examples on how the Figaf Migration Tool handles

SAP Process Integration

to

SAP Cloud Integration Migrations

Overview

The Figaf Migration Tool won the SAP Hack2Build competition by building a tool that can automate migrations from Process Integration to SAP Integration Suite - Cloud Integration.

In this document you will see some examples of how the tool has handled a migration.
The IFlows in the package are generated directly with the Figaf Migration Tool.
Once you have migrated you can test with the Figaf Tool to see what needs to be modified.

Figaf

List of supported migrations

We have made plenty of updates to our tool but we want to remind you of the current features of the Figaf Migration Tool.

Flows

The tool supports the following scenarios:

- Migration of ICO and Classical Receiver Determination. It has been tested on SAP PI 7.11+
- Routing conditions are copied if possible from the configuration
- Synchronous and Asynchronous message process
- A preview of the generated BPMN diagram is shown after the generation
- A list of warnings are created to simplify the process

Mappings

For mapping, we can migrate all mappings even if they use:

- Function libraries. Here the FL will be replaced with a Groovy script. There may be a little work involved in removing special functions like Containers and Dynamic Properties.
- Multi-mappings are handled automatically
- Support Message Types, External Definition, IDOC, and RFC mappings
- RFCLoops is migrated and the user can then insert mappings themselves
- Message Mappings with Parameters

Conditions

- Rules on Receiver and Interface Receiver
- Xpath conditions

- Operation conditions

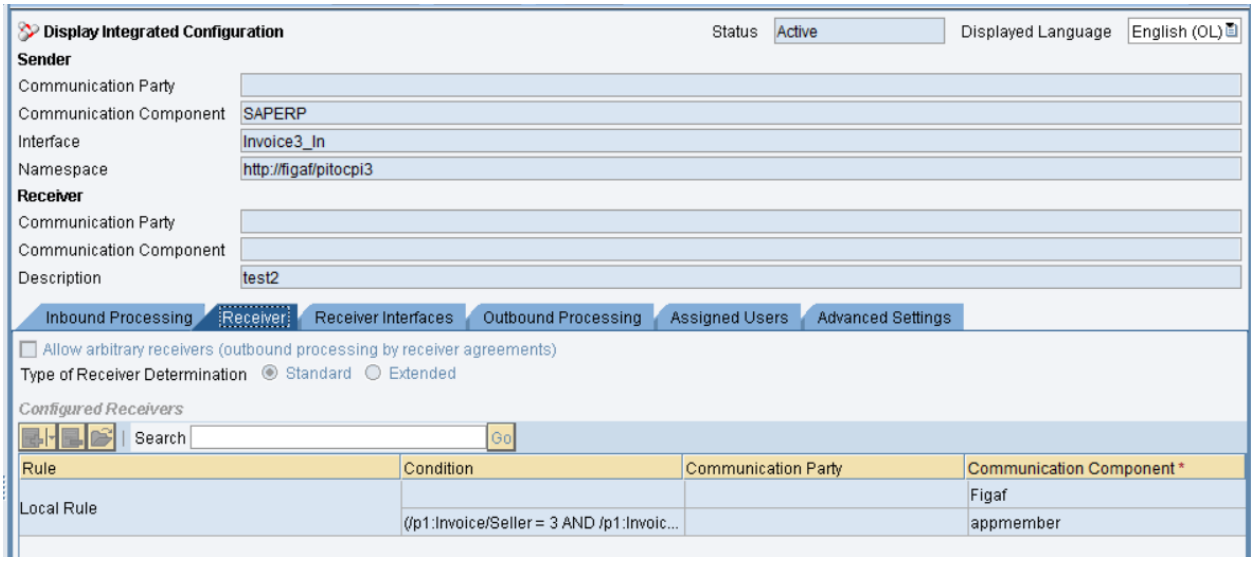
Adapter

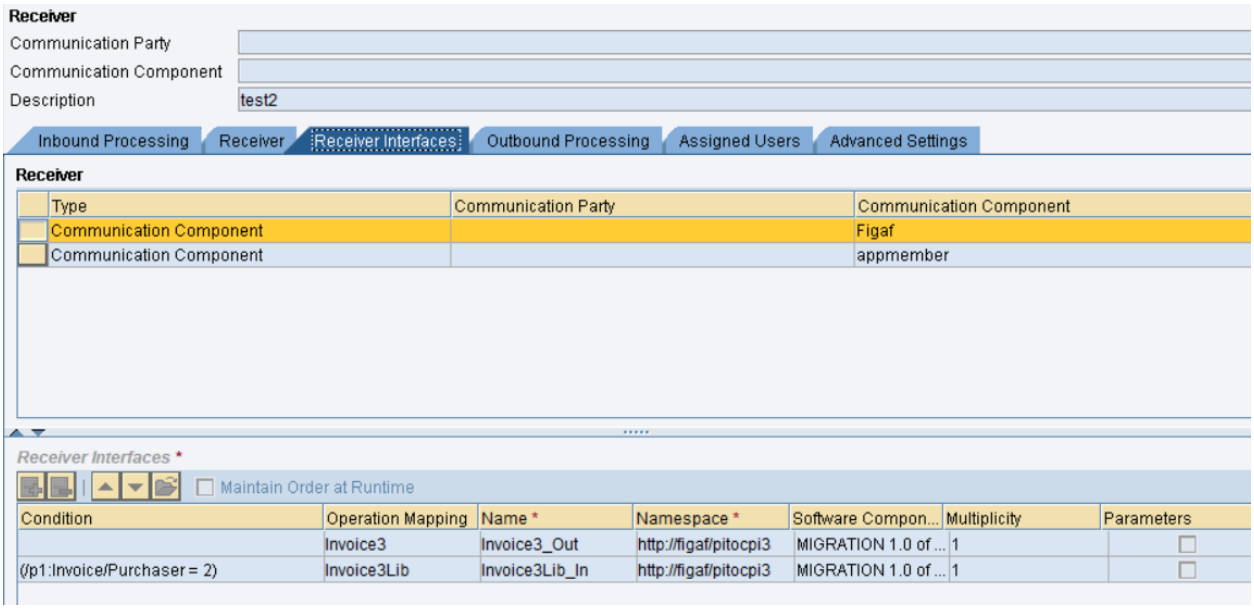
- To convert between a Communication Channel and an SAP CPI block it is possible to use some of the pre-delivered mappings or create your own mapping. It takes a few minutes to create your own custom mapping that matches your requirements.
- You can find the templates in the git repository.

Examples

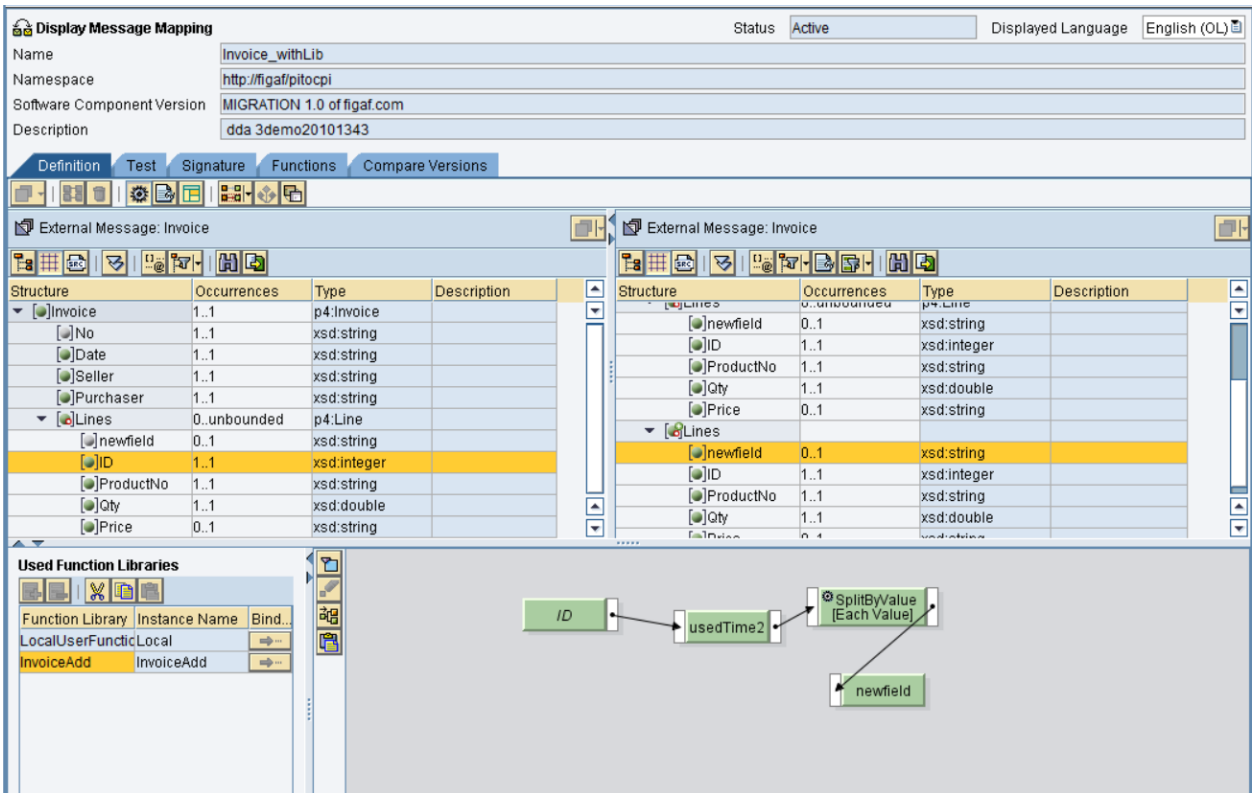
All of the examples in this IFlow have been generated directly using the Figaf Migration Tool.

1. ICO with receiver and interface split with mappings and Function Libraries

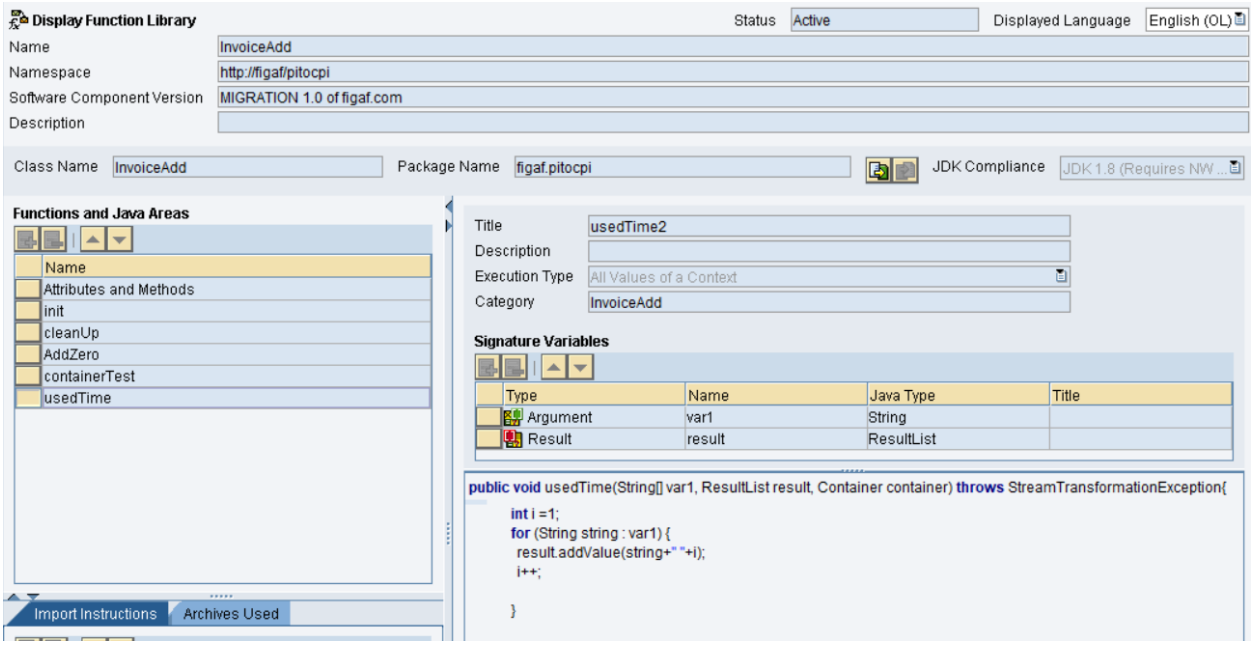




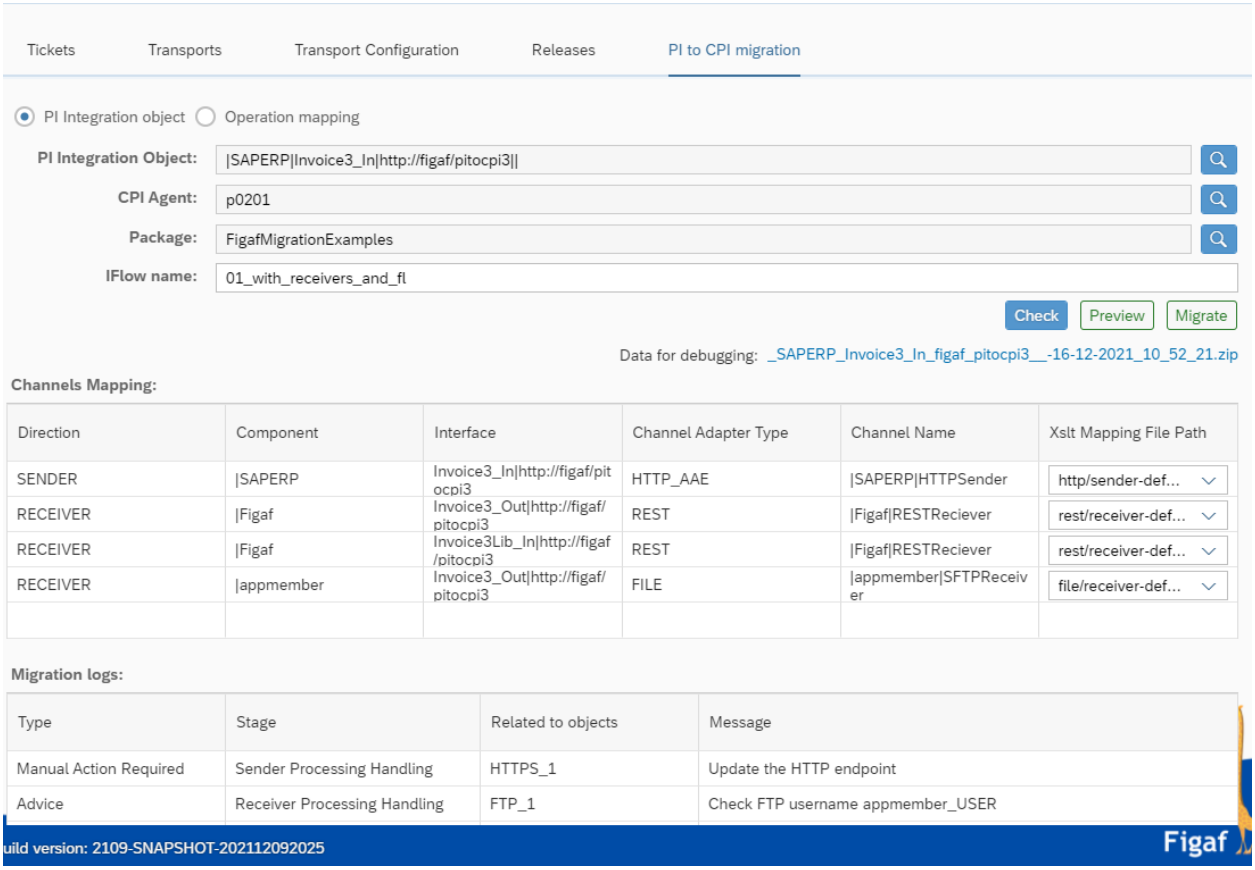
The Message Mapping that uses a Function Library



The Function Library

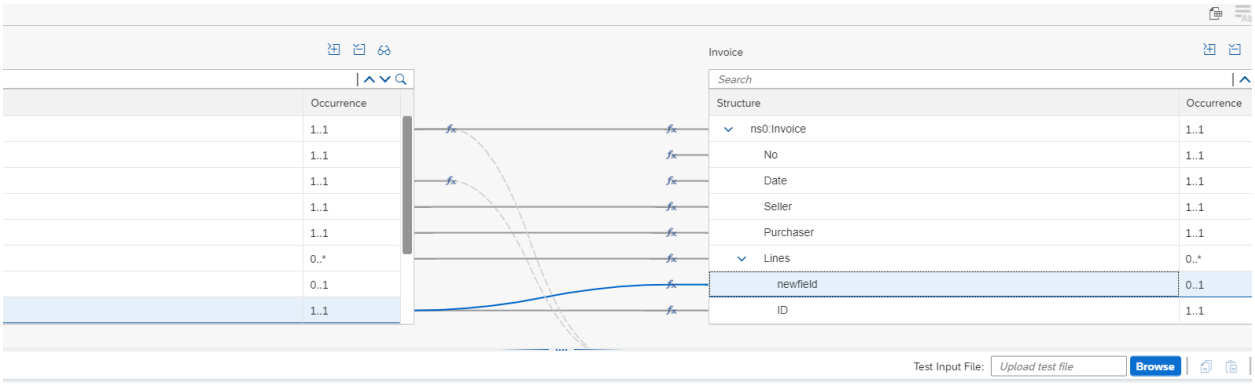


Here is the migration



Figaf

The resulting Message Mapping



The Function Library is added as Groovy Script

Figaf Migration Examples / EX01_Mappings_with_Function_Library / Invoice_wit
MIGRATION_1_0_of_figaf_com_figaf_pitocpi3_InvoiceAdd.grc

```
1  import com.sap.it.api.mapping.*;
2  import java.io.*;
3  import java.lang.reflect.*;
4  import java.util.*;
5
6  def String AddZero(String number, MappingContext context) {
7  int m=17;
8  return "000"+number;
9  }
10 def void usedTime(String[] var1, Output result, MappingContext context) {
11         int i =1;
12         for (String string : var1) {
13             result.addValue(string+" "+i);
14             i++;
15
16         }
17 }
18
```

2. Routing flow

This flow has many different messages to show some of the complicated flows that are begin run.
You can see multi-mappings where the message is split and then processed as

individual messages.
It also contains many different message types to show what options you have.

Sender

Communication Party	
Communication Component	SAPERP
Interface	Invoice01_OA
Namespace	http://figaf/pitocpi

Receiver

Communication Party	
Communication Component	
Description	

Inbound Processing | Receiver | Receiver Interfaces | Outbound Processing | Assigned Users | Advanced Settings

Receiver

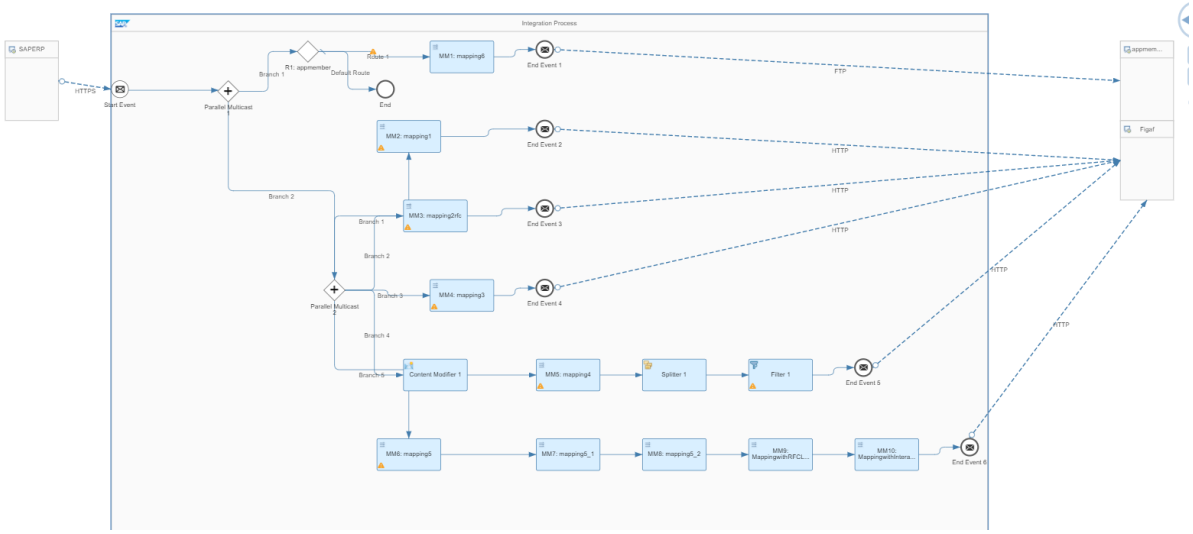
Type	Communication Party	Communication Component
Communication Component		Figaf
Communication Component		appmember

*Receiver Interfaces **

Maintain Order at Runtime

Condition	Operation Mapping	Name *	Namespace *	Software Component ...	Multiplicity	Parameters
	Mapping1	INVOIC.INVOIC02	urn:sap-com:document:	B2B MAPPING KIT 1.0	1	☐
	Mapping2_RFC	SXIDEMO_AIRL_FLIGH	urn:sap-com:document:	SAP BASIS 7.50	1	☐
	Mapping3	Invoice3_In	http://figaf/pitocpi3	MIGRATION 1.0 of fig...	1	☐
		Invoice_In	http://figaf/pitocpi	MIGRATION 1.0 of fig...	1	☐
	Mapping4	Line_In	http://figaf/pitocpi4	MIGRATION 1.0 of fig...	0..unbounded	☐
	Mapping5	ORDERS_INB	http://figaf.com/FigafDual	FIGAFDUALSTACK 1....	1	☑

Resulting IFlow



RFC Lookup

IF a Message Mapping with a RFC Lookup is using an empty shell, then a groovy script for a UDF is created. The user can then implement RFC functions to handle the lookup. An example can be seen at the mapping MappingwithRFCLookup.

Here is the example code that can be implemented to run on the Cloud Foundry system

```
import com.sap.it.api.mapping.*;
import com.sap.conn.jco.JCoDestination;
import com.sap.conn.jco.JCoDestinationManager;
import com.sap.conn.jco.JCoFunction;
import com.sap.conn.jco.JCoParameterList;
import com.sap.conn.jco.JCoRepository;



def void RFC_ZIS_EUCOUNTRY(String[] I_COUNTRYCODE, Output output, MappingContext context) {
  String RFCOutputFieldName = "O_IS_EU_COUNTRY";
  String RFCModuleName = "ZIS_EUCOUNTRY";
//Get RFC destination name from property (Content Modifier)
//Destination name can be found in Cloud Cockpit -> Subaccount -> Destinations
  JCoDestination destination =
JCoDestinationManager.getDestination(context.getProperty("RFCDestination"));
  JCoRepository repo = destination.getRepository();
//Create connection to RFC module in question - in this case 'ZIS_EUCOUNTRY'
  JCoFunction RfcConnection = repo.getFunction(RFCModuleName);
//Build RFC Request.
  JCoParameterList imports = RfcConnection.getImportParameterList()
//setValue(<RFC Input field name>,<value>)
  imports.setValue("I_COUNTRYCODE", I_COUNTRYCODE[0]);


//Call/execute RFC.
  RfcConnection.execute(destination);
```

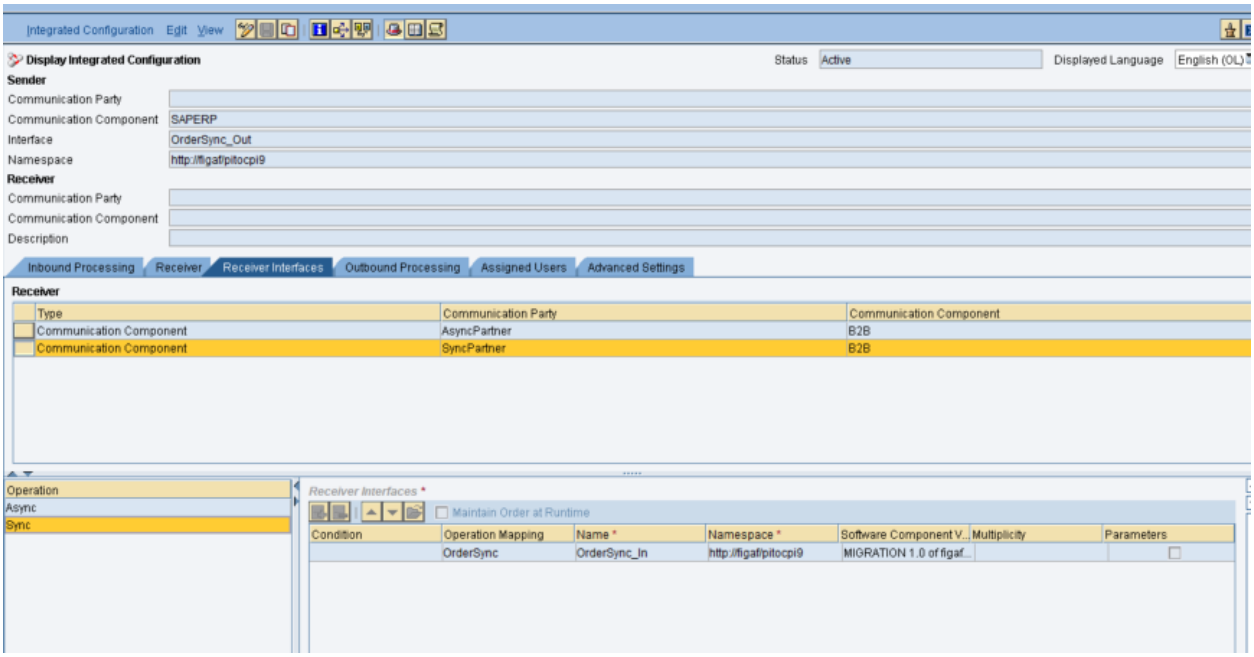
```
//Get RFCResponse fields.
  JCoParameterList exports = RfcConnection.getExportParameterList();
//Get <RFC output Field> = 'O_IS_EU_COUNTRY'
  String retValue = exports.getString(RFCOutputFieldName);
//Add RFC Value to output(mapping) Context
  output.addValue(retValue);


//Add lookup value to log - if logProperty is 'yes'
  String logMe = context.getProperty("logProperty")
  if (logMe.equalsIgnoreCase("yes")) {
     context.setProperty("RFC_" + RFCModuleName + ":" + RFCOutputFieldName, retValue);
  }


}
```

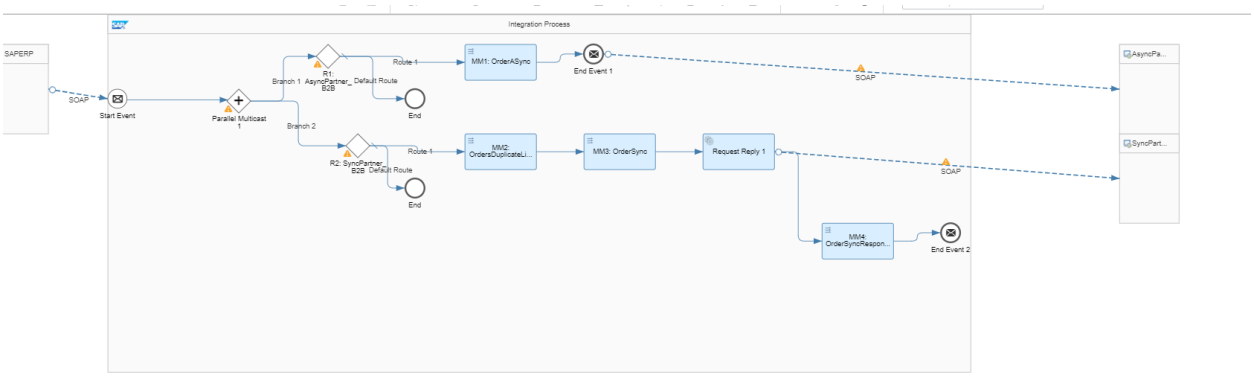
3. Synchronous and Operation Based routing

In this scenario we have added an Operation Condition to the result. The test scenario is Synchronous / BE.



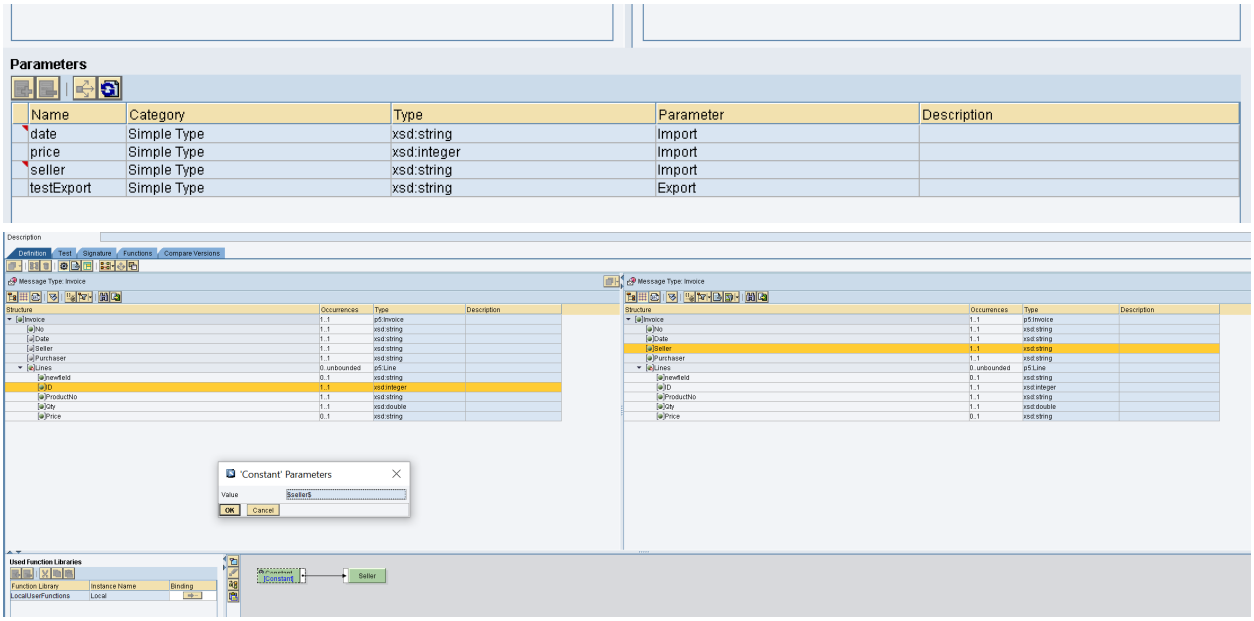
This will result in something like the following:

Notice both request and response mappings are added and the Request Reply is used to handle the request.

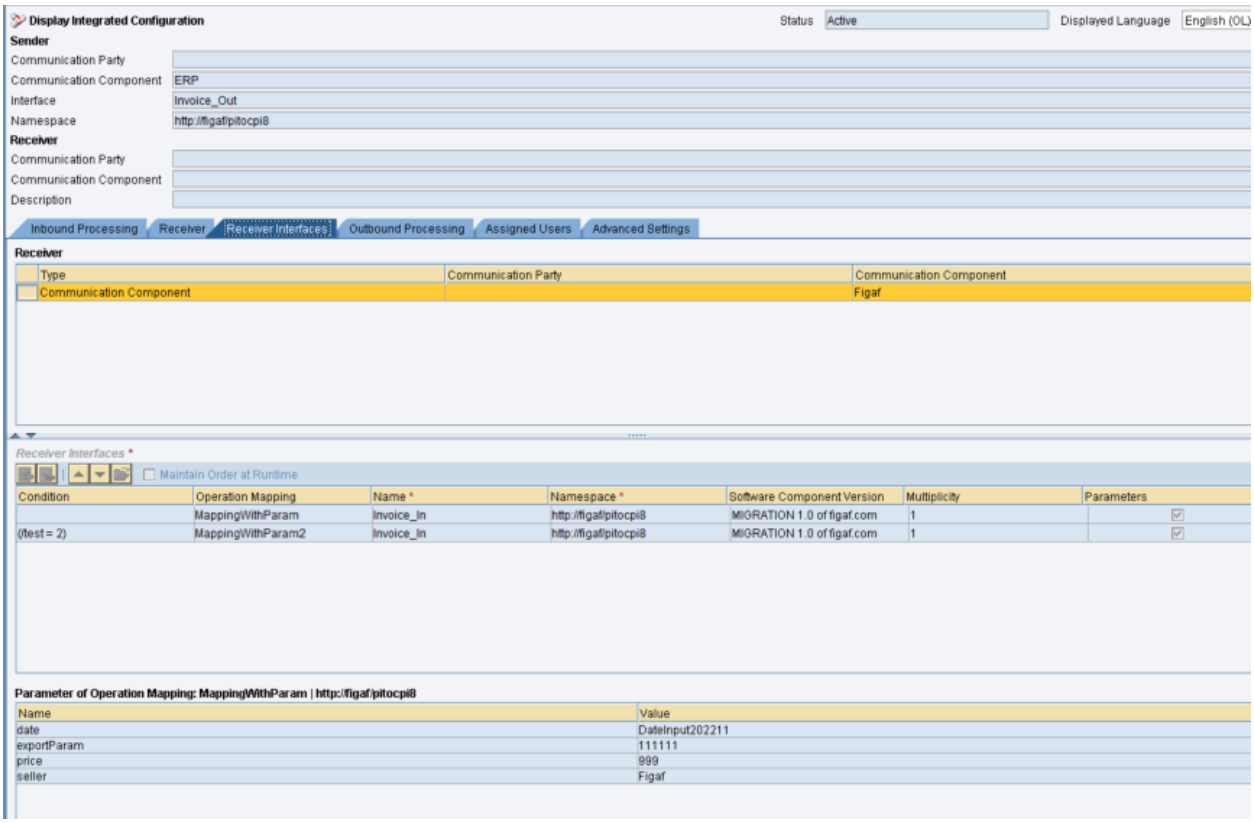


4. Message Mapping with Parameters

Some Message Mappings can have Parameters that are linked via the Operation Mapping to the ICO.

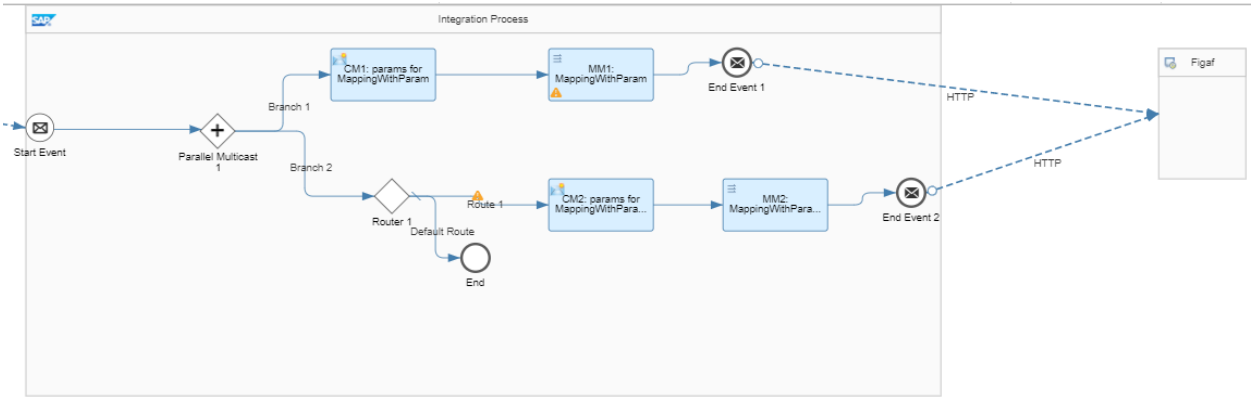


The configuration looks like the following:

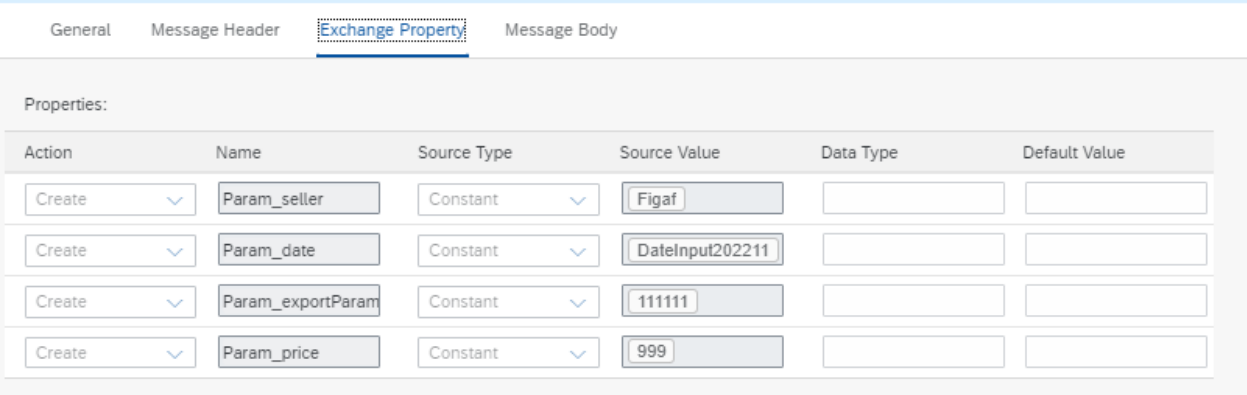


The flow is created to handle the migration.

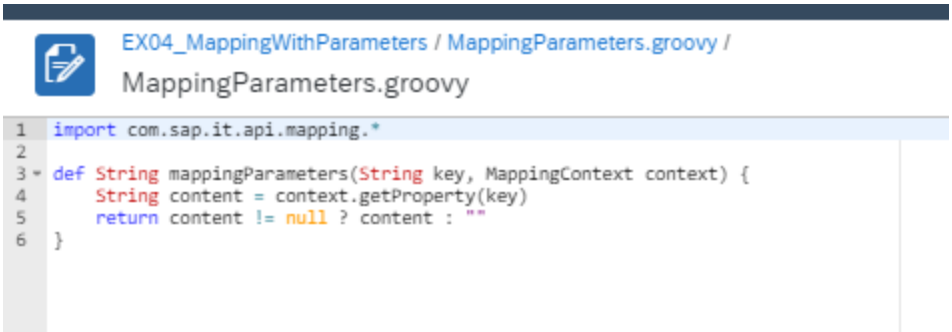
A Content Modifier is added before the mapping to enable the ability to save the configurations from the ICO.



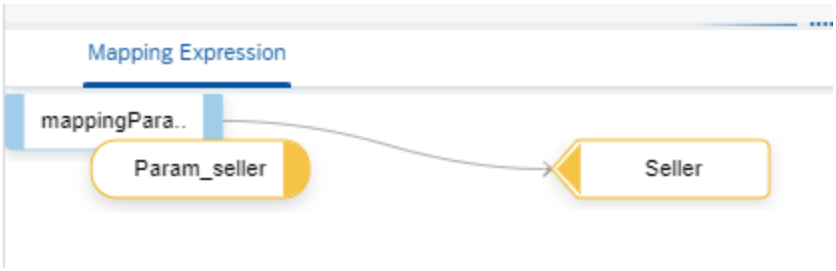
The properties are sent in as an Externalized Parameter and then linked to the parameters.



A simple script is added that can handle the configuration.



The Message Mapping is updated to use the UDF and then the constant is updated.



Try it yourself

The best way to get started is by trying out the Figaf SAP Process Integration to SAP Cloud Integration. We have created a mission at the SAP Discovery Center that guides you through the process.

Start the [mission today]