

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.

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

Display Integrated Configuration Status: Active Displayed Language: English (OL)

Sender

Communication Party:

Communication Component: SAPERP

Interface: Invoice3_In

Namespace: http://figaf/pitocpi3

Receiver

Communication Party:

Communication Component:

Description: test2

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

Allow arbitrary receivers (outbound processing by receiver agreements)

Type of Receiver Determination: Standard Extended

Configured Receivers

Rule	Condition	Communication Party	Communication Component *
Local Rule	/p1:Invoice/Seller = 3 AND /p1:Invoic...		Figaf appmember

Receiver

Communication Party:

Communication Component:

Description: test2

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 Compon...	Multiplicity	Parameters
	Invoice3	Invoice3_Out	http://figaf/pitocpi3	MIGRATION 1.0 of ...	1	<input type="checkbox"/>
(p1:Invoice/Purchaser = 2)	Invoice3Lib	Invoice3Lib_In	http://figaf/pitocpi3	MIGRATION 1.0 of ...	1	<input type="checkbox"/>

The Message Mapping that uses a Function Library

Display Message Mapping Status: Active Displayed Language: English (OL)

Name: Invoice_withLib
 Namespace: http://figaf/pitocpi
 Software Component Version: MIGRATION 1.0 of figaf.com
 Description: dda 3demo20101343

Definition | Test | Signature | **Functions** | Compare Versions

External Message: Invoice

Structure	Occurrences	Type	Description
Invoice	1..1	p4:Invoice	
No	1..1	xsd:string	
Date	1..1	xsd:string	
Seller	1..1	xsd:string	
Purchaser	1..1	xsd:string	
Lines	0..unbounded	p4:Line	
newfield	0..1	xsd:string	
ID	1..1	xsd:integer	
ProductNo	1..1	xsd:string	
Qty	1..1	xsd:double	
Price	0..1	xsd:string	

External Message: Invoice

Structure	Occurrences	Type	Description
newfield	0..1	xsd:string	
ID	1..1	xsd:integer	
ProductNo	1..1	xsd:string	
Qty	1..1	xsd:double	
Price	0..1	xsd:string	

Used Function Libraries

Function Library	Instance Name	Bind...
LocalUserFuncLocal		
InvoiceAdd	InvoiceAdd	

```

graph LR
  ID[ID] --> usedTime2[usedTime2]
  usedTime2 --> SplitByValue[SplitByValue [Each Value]]
  SplitByValue --> newfield[newfield]
  
```

The Function Library

Display Function Library Status: **Active** Displayed Language: **English (OL)**

Name: InvoiceAdd
 Namespace: http://figaf/pitocpi
 Software Component Version: MIGRATION 1.0 of figaf.com
 Description:

Class Name: InvoiceAdd Package Name: figaf.pitocpi JDK Compliance: JDK 1.8 (Requires NW ...)

Functions and Java Areas

- Name
- Attributes and Methods
- init
- cleanUp
- AddZero
- containerTest
- usedTime

Title: usedTime2
Description:
Execution Type: All Values of a Context
Category: InvoiceAdd

Signature Variables

Type	Name	Java Type	Title
Argument	var1	String	
Result	result	ResultList	

```

public void usedTime(String[] var1, ResultList result, Container container) throws StreamTransformationException{
    int i =1;
    for (String string : var1){
        result.addValue(string+"*"+i);
        i++;
    }
}

```

Here is the migration

Tickets Transports Transport Configuration Releases **PI to CPI migration**

PI Integration object Operation mapping

PI Integration Object: [SAPERP|Invoice3_In|http://figaf/pitocpi3|]

CPI Agent: p0201

Package: FigafMigrationExamples

IFlow name: 01_with_receivers_and_ft


Data for debugging: [_SAPERP_Invoice3_In_figaf_pitocpi3_-16-12-2021_10_52_21.zip](#)

Channels Mapping:

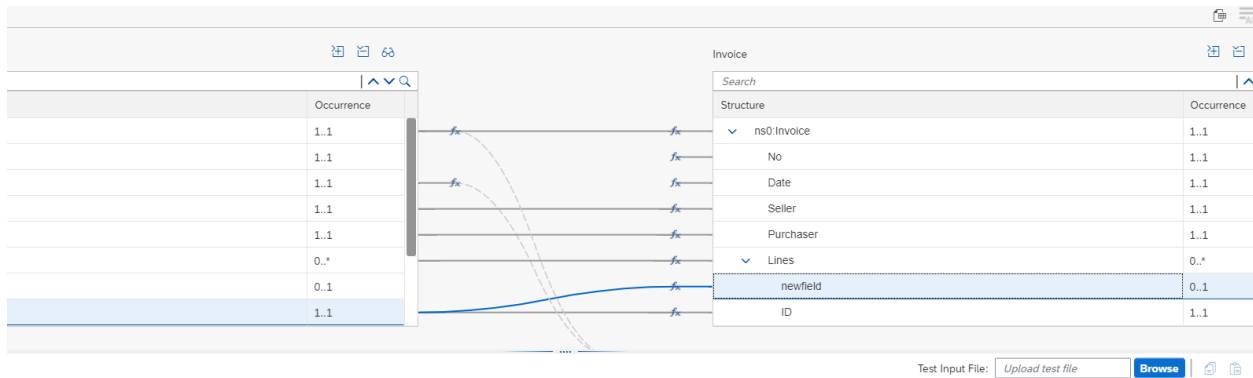
Direction	Component	Interface	Channel Adapter Type	Channel Name	Xslt Mapping File Path
SENDER	SAPERP	Invoice3_In http://figaf/pitocpi3	HTTP_AAE	SAPERP HTTPSender	http/sender-def... <input type="button" value="v"/>
RECEIVER	Figaf	Invoice3_Out http://figaf/pitocpi3	REST	Figaf RESTReceiver	rest/receiver-def... <input type="button" value="v"/>
RECEIVER	Figaf	Invoice3Lib_In http://figaf/pitocpi3	REST	Figaf RESTReceiver	rest/receiver-def... <input type="button" value="v"/>
RECEIVER	appmember	Invoice3_Out http://figaf/pitocpi3	FILE	appmember SFTPReceiver	file/receiver-def... <input type="button" value="v"/>

Migration logs:

Type	Stage	Related to objects	Message
Manual Action Required	Sender Processing Handling	HTTPS_1	Update the HTTP endpoint
Advice	Receiver Processing Handling	FTP_1	Check FTP username appmember_USER

uid version: 2109-SNAPSHOT-202112092025 

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

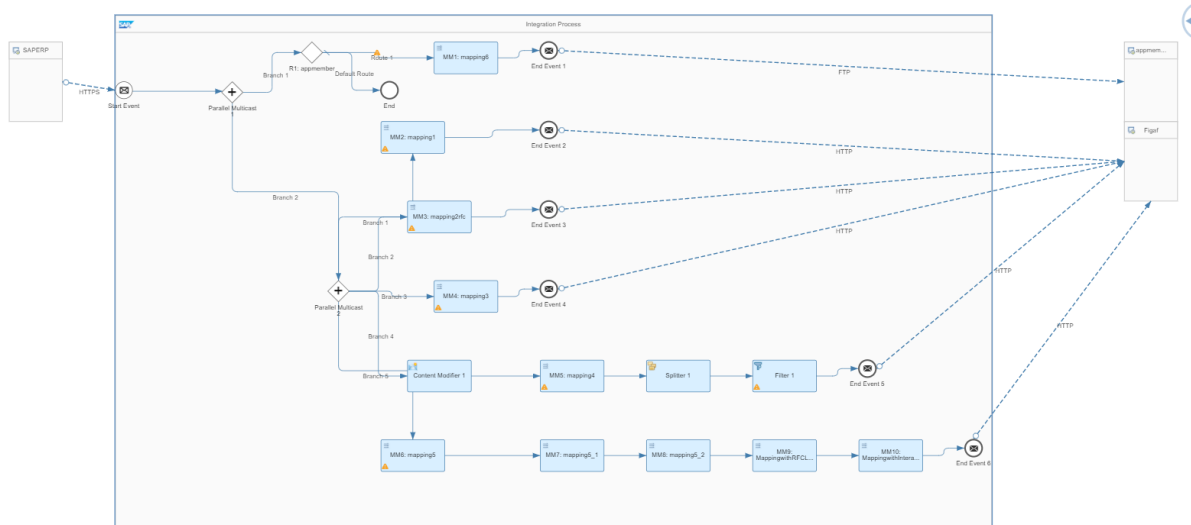
Type	Communication Party	Communication Component
Communication Component	Figaf	Figaf
Communication Component		appmember

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	<input type="checkbox"/>
	Mapping2 RFC	SXIDEMO_AIRL_FLIGH	urn:sap-com:document	SAP BASIS 7.50	1	<input type="checkbox"/>
	Mapping3	Invoice3_In	http://figaf/pitocpi3	MIGRATION 1.0 of fig...	1	<input type="checkbox"/>
	Mapping4	Invoice_In	http://figaf/pitocpi	MIGRATION 1.0 of fig...	1	<input type="checkbox"/>
	Mapping5	Line_In	http://figaf/pitocpi4	MIGRATION 1.0 of fig...	0..unbounded	<input type="checkbox"/>
	Mapping5	ORDERS_INB	http://figaf.com/figafDual	FIGAFDUALSTACK 1....	1	<input checked="" type="checkbox"/>

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.

The screenshot displays the SAP Integrated Configuration interface. The top section shows the configuration for a Sender (SAPERP) and Receiver. The Receiver section is expanded to show a table of communication components:

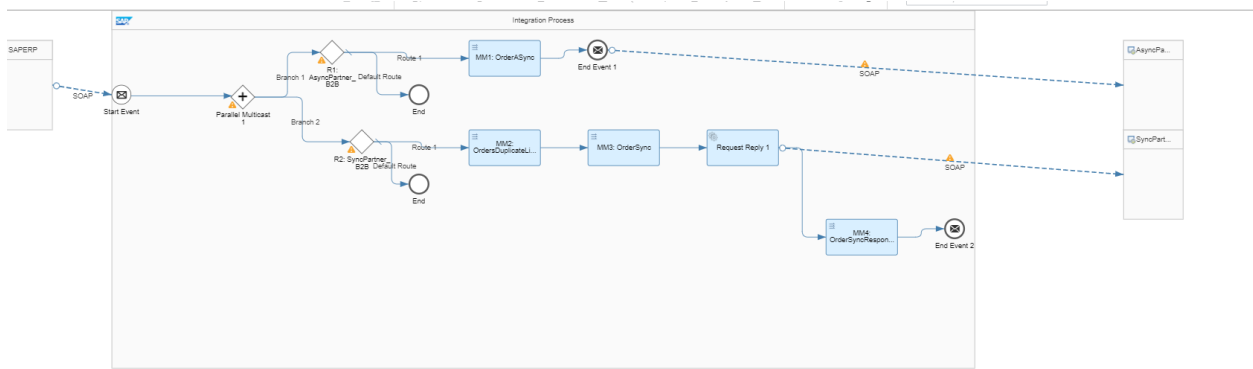
Type	Communication Party	Communication Component
Communication Component	AsyncPartner	B2B
Communication Component	SyncPartner	B2B

Below this, the 'Receiver Interfaces' section is visible, showing a table of operation mappings:

Condition	Operation Mapping	Name *	Namespace *	Software Component V...	Multiplicity	Parameters
	OrderSync	OrderSync_In	http://igaf/pitocpi9	MIGRATION 1.0 of igaf...		<input type="checkbox"/>

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 Parameters table is as follows:

Name	Category	Type	Parameter	Description
date	Simple Type	xsd:string	Import	
price	Simple Type	xsd:integer	Import	
seller	Simple Type	xsd:string	Import	
testExport	Simple Type	xsd:string	Export	

The Message Mapping configuration shows two message types: Invoice and Invoice. A 'Constant Parameters' dialog box is visible, showing a dropdown menu with 'Seller' selected. The dialog has 'OK' and 'Cancel' buttons.

The configuration looks like the following:

Display Integrated Configuration Status: Active | Displayed Language: English (OL)

Sender

Communication Party: _____
 Communication Component: ERP
 Interface: Invoice_Out
 Namespace: http://figafpitocpi8

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

Receiver Interfaces *

Maintain Order at Runtime

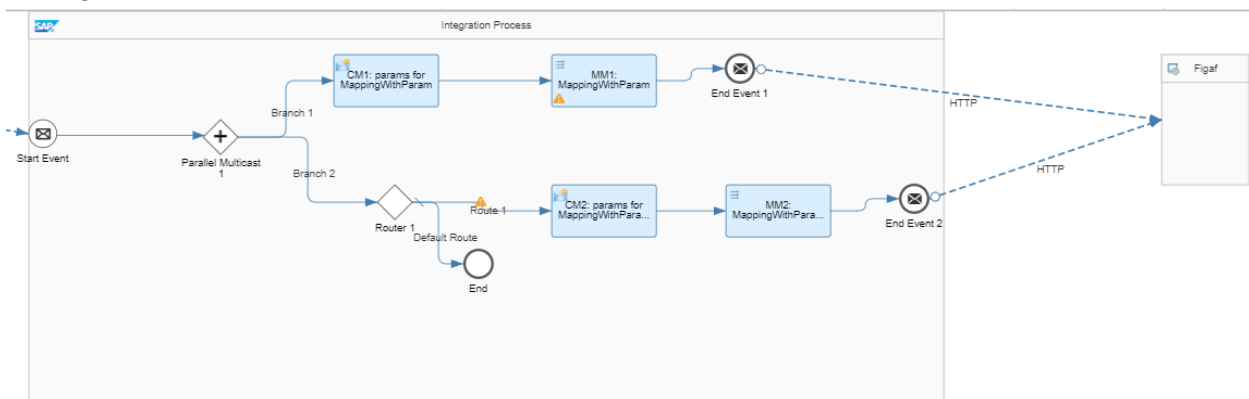
Condition	Operation Mapping	Name *	Namespace *	Software Component Version	Multiplicity	Parameters
	MappingWithParam	Invoice_In	http://figafpitocpi8	MIGRATION 1.0 of figaf.com	1	<input checked="" type="checkbox"/>
(test = 2)	MappingWithParam2	Invoice_In	http://figafpitocpi8	MIGRATION 1.0 of figaf.com	1	<input checked="" type="checkbox"/>

Parameter of Operation Mapping: MappingWithParam | http://figafpitocpi8

Name	Value
date	DateInput202211
exportParam	111111
price	999
seller	Figaf

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.

General Message Header <u>Exchange Property</u> Message Body					
Properties:					
Action	Name	Source Type	Source Value	Data Type	Default Value
Create ▾	Param_seller	Constant ▾	Figaf		
Create ▾	Param_date	Constant ▾	DateInput202211		
Create ▾	Param_exportParam	Constant ▾	111111		
Create ▾	Param_price	Constant ▾	999		

A simple script is added that can handle the configuration.

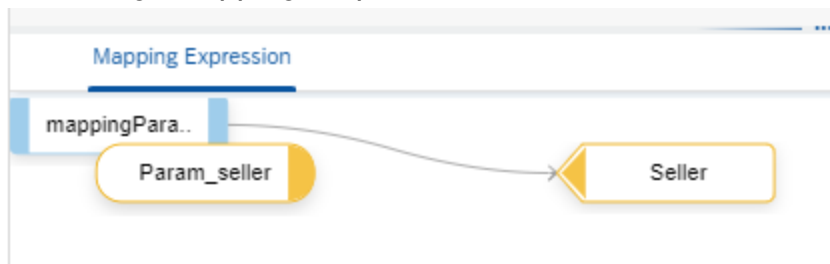
EX04_MappingWithParameters / MappingParameters.groovy / MappingParameters.groovy

```

1 import com.sap.it.api.mapping.*
2
3 def String mappingParameters(String key, MappingContext context) {
4     String content = context.getProperty(key)
5     return content != null ? content : ""
6 }

```

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](#)

The screenshot shows a mission page titled "Automate the Migration from SAP Process Integration to SAP Cloud Integration with the Figaf Migration Tool". The page includes a "Start Mission" button and a "More" dropdown. Below the title, there are navigation links for "Overview", "Project Board", "Resources", "Support", and "Related Missions". A note states: "To access the card content, please click on the 'Start Mission' button." The mission is divided into four phases: DISCOVER, PREPARE, SET UP, and COMPLETE. Each phase contains several tasks with associated tags like "BASIC", "IMPORTANT", "ADVANCED", "MILESTONE", "OPTIONAL", "DELIVERABLE", "LINK", "READ", "DEEP", "COLLABORATE", and "MUST".

Home / Missions

Automate the Migration from SAP Process Integration to SAP Cloud Integration with the Figaf Migration Tool [Start Mission](#) [More](#)

FIGAF

Overview **Project Board** Resources Support Related Missions

To access the card content, please click on the 'Start Mission' button.

DISCOVER

- BASIC** **LINK**
Check how your migrations can be automated.
- LINK** **READ**
Read blog post about updates to our migration tool
- IMPORTANT** **DEEP**
Learn about the Figaf Migration Tool
- IMPORTANT** **LINK**
Install and test the Figaf Migration Tool with a trial license
- DELIVERABLE** **COLLABORATE**
Perform fit-gap analysis and collect requirements

PREPARE

- IMPORTANT**
Scope out project
- ADVANCED**
Review security and IT governance
- LINK** **COLLABORATE**
Contact our experts and schedule a meeting to conduct a solution walkthrough
- LINK** **MILESTONE**
Set up the contract, pricing, and procurement

SET UP

- ADVANCED**
Procure and set up hardware requirements
- DELIVERABLE** **MILESTONE**
Install and configure the Figaf Migration Tool with your procured license
- COLLABORATE** **MUST**
Onboarding and training
- DELIVERABLE**
Execute final testing

COMPLETE

- MILESTONE** **OPTIONAL**
Go live and celebrate
- DELIVERABLE**
Commence bulk migration processes after "Go-live date"
- ADVANCED**
Apply continuous improvements and upgrades
- BASIC** **DELIVERABLE**
Share your success and select the next mission
- LINK** **MILESTONE**
Complete the mission