

[PROXY](#)

POSTED ON AUGUST 1, 2019 BY ISURU FERNANDO



There are two methods to debug ABAP inbound proxies in SAP.

1. Debugging using SPROXY test feature
2. External Debugging

Each method has its own advantages, disadvantages, limitations and use cases. But, using these two methods you can fully debug and test an inbound ABAP proxies.

In this article we will discuss in detail how to perform each debugging and testing method and their use cases and advantages.

[Privacy](#) - [Terms](#)



way to test and debug SAP application processing without having to connect with the external system or PI/PO. Moreover, using this method ABAP developers can work independently to complete **Developer Unit Testing** (DUT) scenarios without integrating the interface end-to-end.

Use Cases – Troubleshoot using External Debugging

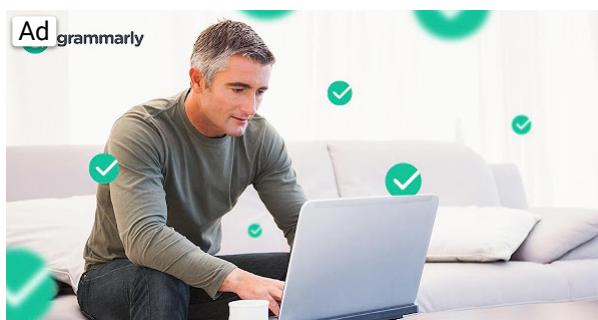
We can use the **external debugging** method if you need to debug the proxy logic when messages are triggered by the external system or PI/PO. This will allow us to **troubleshoot connectivity** related issues, issues due to **proxy version inconsistencies**, **character set issues**, etc. Or if proxy messages are sent **with attachments** you can use external debugging methods to identify

proxy connectivity between ESB and SAP HANA completed and proxy class generated/activated.

Besides, you can check my previous article on how to implement an [Inbound Proxy interface](#) to update exchange rates in SAP.

How to Find the ABAP Proxy Implementing Class:

Before you can debug the ABAP inbound proxy, first you must **find the ABAP proxy class**. Then, you need to **set a break-point** in the appropriate method or ABAP code line in the ABAP implementing class. Break-point could be either a **session break-point** or an **external break-point** depending on the debugging method you choose.



Correct All Grammar Errors

Easily fix typos, grammatical mistakes, and other common issues before you hit Send

 Grammarly

LEARN MORE





- Step 2 – Determine the **ABAP Proxy class** using **SPROXY** transaction

Step 1 – Find the Inbound Service Interface and Namespace



- Option 1 – Find the Inbound Service Interface via the Integrated Configuration Object (ICO):
- Option 2 – Find Interface using [iFlow](#) in NWDS:

Option 1 – Find the Inbound Service Interface via ICO:

To find the Inbound Service Interface from ICO, open the Integration Directory (ID) and navigate to ICO.

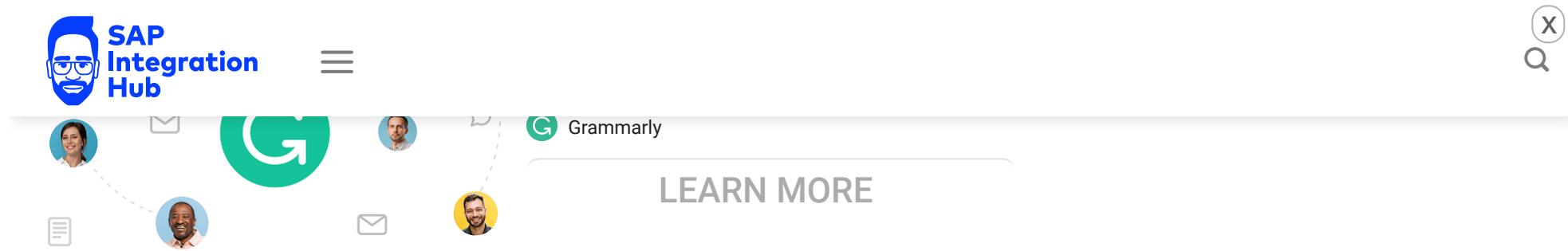
Then, go to the “Receiver Interface” tab of ICO to find the Inbound Service Interface name. In this tab, you can also find the Namespace and the Software Component Version (SWCV) of the Inbound Interface. This information is required for you to find the ABAP Proxy class in the SAP back-end system in SPROXY transaction.

The screenshot shows the SAP Integration Hub interface. On the left, there's a sidebar with various configuration items like Receiver Rule, Sender Agreement, and Integrated Configuration. Under Integrated Configuration, there's a list of interfaces, with 'Test_A | ExchangeRates_Out_Async' highlighted by a red box and a green circle labeled '1'. The main panel shows a communication party setup for 'Test_A' with an interface named 'ExchangeRates_Out_Async' and a namespace 'urn:Source_System:ExchangeRates'. Below this, there's a 'Receiver' section with its own communication party and a description 'Generated for dir://IFLOW/001/ChangeRateswithFaultMessage'. The top navigation bar has tabs for Inbound Processing, Receiver, Receiver Interfaces (which is active and highlighted by a red box and green circle '2'), Outbound Processing, Assigned Users, and Advanced Settings. The 'Receiver' tab is also visible. At the bottom, there's a detailed view of the 'Receiver Interfaces' table, showing a single row with a condition 'ExchangeRates_to_ExchangeRateswithFaultMessage' and an operation mapping 'ExchangeRateswithFaultMessage_Inb_Async' in the 'Name' field, which is also highlighted by a red box and green circle '3'.

Receiver Interface in ICO in ESR

Option 2 – Find Interface using iFlow in NWDS:

If you have implemented the interface using [iFlow](#), we can also find the Interface detail directly via the iFlow.

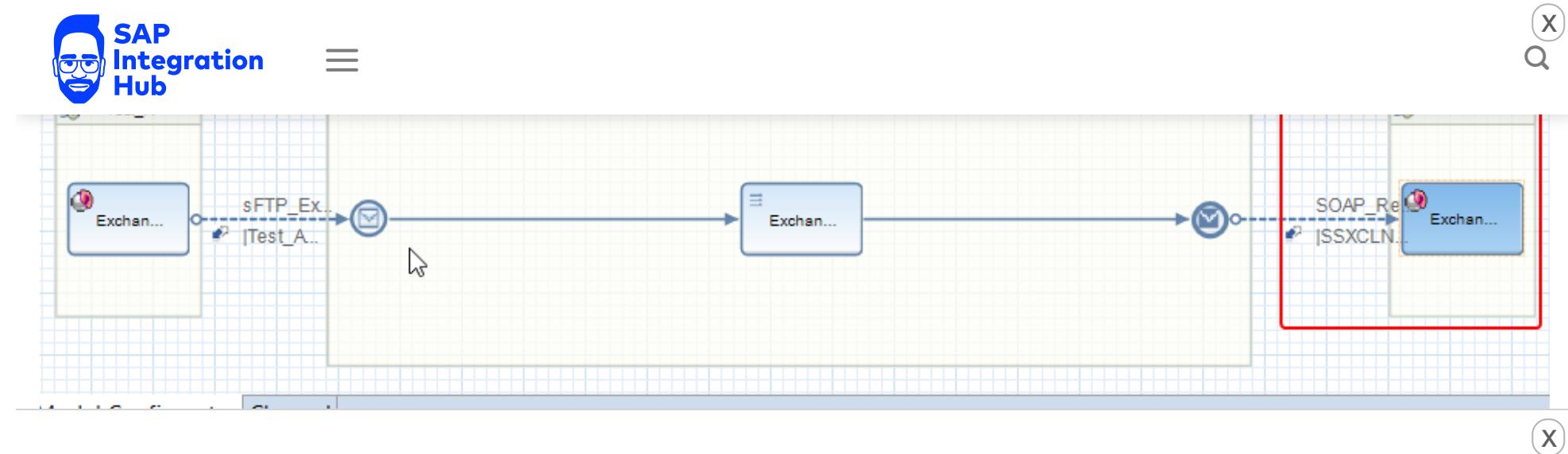


Open [NWDS](#) and navigate to your iFlow under the Integration Flow navigation menu.

The screenshot shows the SAP Web IDE (NWDS) interface. At the top, there are tabs for 'PI Explorer', 'Enterprise ...', and the currently active 'ExchangeRateswithFaultMessage'. On the left, a navigation menu lists 'ExchangeRates', 'ExchangeRateswithFaultMessage' (which is highlighted with a red box), 'Receiver Rules', and 'Systems'. On the right, a detailed view of an integration flow is shown. It features a central box labeled 'Test_A' containing a 'Exchange...' component. An outgoing arrow from this component is labeled 'sFTP_Ex...' and points to a 'Test_A...' message icon. The entire interface has a light blue grid background.

iFlow navigation menu in NWDS

Then, click on the receiver interface name to reveal the detail of the Inbound Service Interface. Here you can find the Name, Namespace, and SWCV of the Inbound Interface.



Interface detail (name, namespace and SWCV) in iFlow

Step 2 – Determine the ABAP Proxy Class Using SPROXY Transaction

Now that you have identified the Inbound Service Interface details such as name, SWCV, and Namespace, go to transaction SPROXY in SAP back-end system and navigate to the interface using the navigation tree.

In the navigation tree select of SPROXY, select **Source > ESR > SWCVs > SWCV > Namespaces > Namespace > Object Types > Service Providers > Service Interface Name**.



Privacy - Terms



Source	Enterprise Service Repository
ESR	Enterprise Service Repository
SWCs	
INTERFACEDEMO INTERFACEDEMO, 1.0 of pocasts.com	INTERFACEDEMO, 1.0 of pocasts.com
MAPPING MAPPING, 1.0 of integrationhub.com	MAPPING, 1.0 of integrationhub.com
SENDERSYSTEM SENDERSYSTEM, 1.0 of integrationhub.com	SENDERSYSTEM, 1.0 of integrationhub.com
SOURCE_SYSTEM SOURCE_SYSTEM, 1.0 of integrationhub.com	SOURCE_SYSTEM, 1.0 of integrationhub.com
SWCV_APTOS SWCV_APTOS 1.0 of trg.com	SWCV_APTOS 1.0 of trg.com
SWCV_AVERY SWCV_AVERY 1.0 of averydennison.com	SWCV_AVERY 1.0 of averydennison.com
SWCV_BILLTRUST SWCV_BILLTRUST 1.0 of billtrust.com	SWCV_BILLTRUST 1.0 of billtrust.com
SWCV_MAPPING SWCV_MAPPING, 1.0 of mapping	SWCV_MAPPING, 1.0 of mapping
SWCV_THIRDPARTY_1 SWCV_THIRDPARTY_1, 1.0 of thirdparty-vendor-1	SWCV_THIRDPARTY_1, 1.0 of thirdparty-vendor-1



urn:Target_System:ProductDetail	
urn:Target_System:DecodedString	
urn:Target_System:Base64EncodedString	
urn:Target_System:Base64OutputMessage	
urn:Target_System:ExchangeRates	
Object Types	3
Data Types	
Fault Message Types	
Service Providers	4
ExchangeRates_Inb_Async	urn:Target_System:ExchangeRates
ExchangeRateswithFaultMessage_Inb_Async	urn:Target_System:ExchangeRates
Message Types	5
Objects	
ESR (local)	Enterprise Service Repository (backend view)
External	Created from external WSDL/Schema
Backend	Modelled locally in the backend system



Navigation menu of SPROXY in SAP back-end system

Double click of the Service Interface Name under the Service Provider view, to find the SAP ABAP OO Implementing class of interface in "Implementing Class" field.

Privacy - Terms



The screenshot shows the SAP Integration Hub interface. At the top left is the SAP Integration Hub logo. To the right are three horizontal menu bars. On the far right are two circular icons: one with an 'X' and one with a magnifying glass.

The main area displays two tables. The top table is titled "Service Provider" and contains the following data:

Name	ExchangeRateswithFaultMessage_Inb_Async
Namespace	urn:Target_System:ExchangeRates
ABAP Object	INTF Interface
ABAP Name	ZII_EXCHANGE_RATESWITH_FAULT_M
Prefix	
Source	Enterprise Services Repository
Description	Proxy Interface (generated)

The "Description" field is highlighted with a red border.

The bottom table is titled "Implementation Class of SAP ABAP Proxy" and contains the following data:

Original Language	EN English			
Created by	ISURUF	on	13.05.2019	12:15:01
Changed by	ISURUF	on	13.05.2019	12:15:13

Implementation Class of SAP ABAP Proxy

Debugging ABAP Proxy Using SPROXY

As the first method, let's look at how to debug ABAP proxies using transaction SPROXY.

Step 1 – Go to Service Interface in SPROXY

As we discussed in the previous chapter, go the inbound Service Interface in the transaction SPROXY. You can use the navigation tree menu of SPROXY transaction to find your Service Interface and Proxy Class.

The screenshot shows two tables representing service definitions in the SAP Integration Hub.

Table 1: Enterprise Service Repository (ESR)

Source	Enterprise Service Repository
ESR	INTERFACEDEMO, 1.0 of podcasts.com
SWCs	MAPPING, 1.0 of integrationhub.com
INTERFACEDEMO INTERFACEDEMO, 1.0 of podcasts.com	SENDERSYSTEM SENDERSYSTEM, 1.0 of integrationhub.com
MAPPING MAPPING, 1.0 of integrationhub.com	SOURCE_SYSTEM SOURCE_SYSTEM, 1.0 of integrationhub.com
SENDERSYSTEM SENDERSYSTEM, 1.0 of integrationhub.com	SWCV_APTOS SWCV_APTOS 1.0 of trg.com
SOURCE_SYSTEM SOURCE_SYSTEM, 1.0 of integrationhub.com	SWCV_AVERY SWCV_AVERY 1.0 of averydennison.com
SWCV_APTOS SWCV_APTOS 1.0 of trg.com	SWCV_BILLTRUST SWCV_BILLTRUST 1.0 of billtrust.com
SWCV_AVERY SWCV_AVERY 1.0 of averydennison.com	SWCV_MAPPING SWCV_MAPPING, 1.0 of mapping
SWCV_BILLTRUST SWCV_BILLTRUST 1.0 of billtrust.com	SWCV_THIRDPARTY_1 SWCV_THIRDPARTY_1, 1.0 of thirdparty-vendor-1
SWCV_MAPPING SWCV_MAPPING, 1.0 of mapping	
SWCV_THIRDPARTY_1 SWCV_THIRDPARTY_1, 1.0 of thirdparty-vendor-1	

Table 2: Target System

urn:Target_System:ProductDetail	
urn:Target_System:DecodedString	
urn:Target_System:Base64EncodedString	
urn:Target_System:Base64OutputMessage	
urn:Target_System:ExchangeRates	
Object Types	3
Data Types	
Fault Message Types	
Service Providers	4
ExchangeRates_Inb_Async	urn:Target_System:ExchangeRates
ExchangeRateswithFaultMessage_Inb_Async	urn:Target_System:ExchangeRates
Message Types	
Objects	
ESR (local)	Enterprise Service Repository (backend view)
External	Created from external WSDL/Schema
Backend	Modelled locally in the backend system

Navigation menu of SPROXY in SAP back-end system

Step 2 – Set a Session Debugging Break-Point in ABAP Proxy Method.

Go to ABAP proxy method by double-clicking on the "Implementing class".



Service Provider	
Name	ExchangeRateswithFaultMessage_Inb_Async
Namespace	urn:Target_System:ExchangeRates
ABAP Object	INTF Interface
ABAP Name	ZII_EXCHANGE_RATESWITH_FAULT_M
Prefix	
Source	Enterprise Services Repository
Description	Proxy Interface (generated)

Original Language	EN English			
Created by	ISURUF	on	13.05.2019	12:15:01
Changed by	ISURUF	on	13.05.2019	12:15:13

Implementing class of Service Provider in SAP back-end

Then navigate to the instance method of the Proxy class.

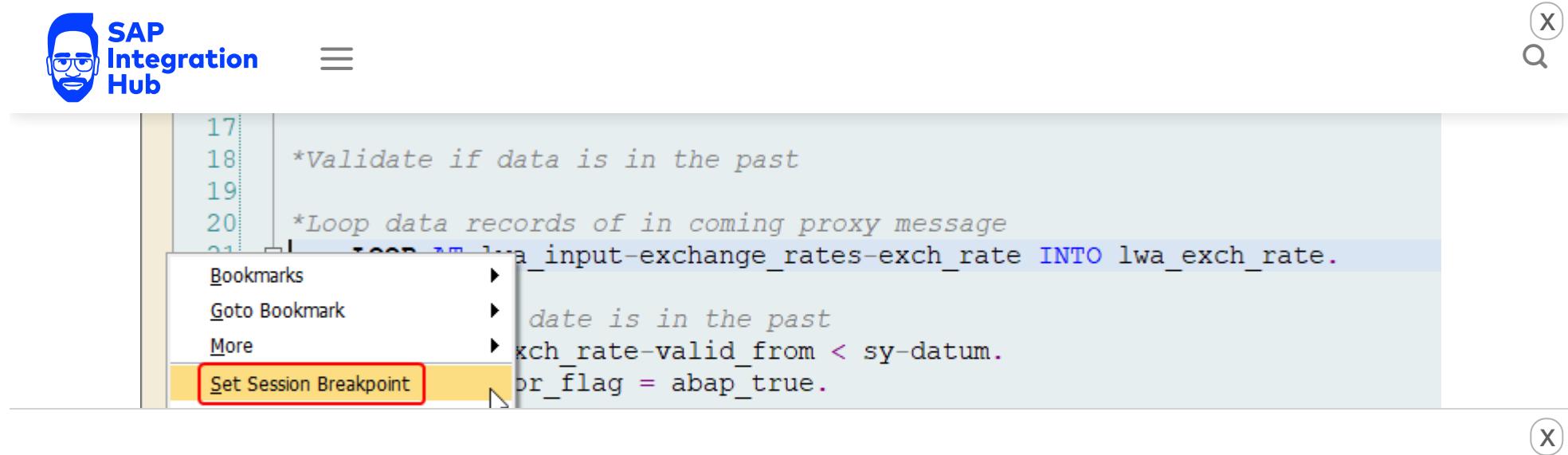


A screenshot of the SAP Integration Hub interface showing a table of methods for an ABAP class. The table has columns for Method, Level, Visibility, and others. One row is selected, showing the method `ZII_EXCHANGE_RATESWITHFAULT_M~EXCHANGE_RATESWITHFAULT_MESSA`.

Method	Level	Visibility	M...	D...
<code>ZII_EXCHANGE_RATESWITHFAULT_M~EXCHANGE_RATESWITHFAULT_MESSA</code>	Instance	Method	Public	

Method of Implementing ABAP class of the Service Provider

Set a session break-point in an appropriate line of the ABAP proxy code. You can do this by right-clicking on the left side of the code editor window.



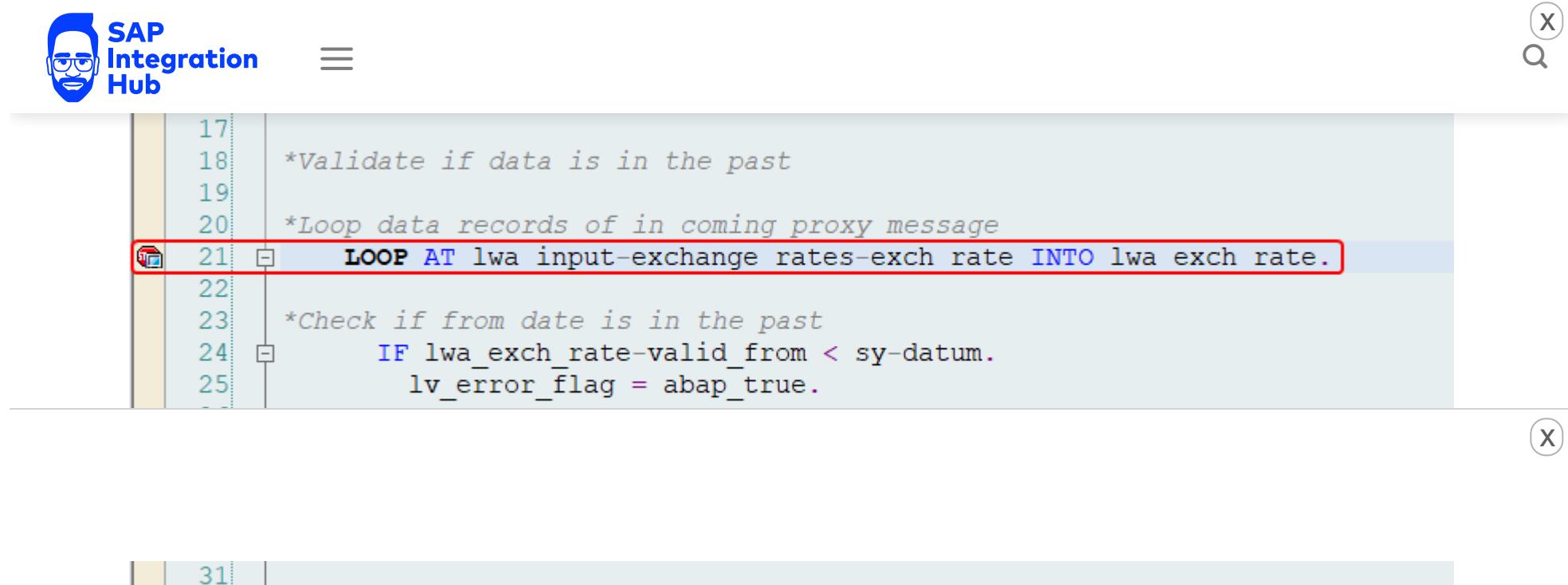
The screenshot shows the SAP Integration Hub interface with an ABAP editor open. A context menu is displayed over a line of code. The menu items are:

- Bookmarks
- Goto Bookmark
- More
- Set Session Breakpoint** (highlighted with a red box)

The code in the editor is:

```
17 *Validate if data is in the past
18
19
20 *Loop data records of in coming proxy message
21 LOOP AT lwa_input-exchange_rates-exch_rate INTO lwa_exch_rate.
22   IF lwa_exch_rate-valid_from < sy-datum.
23     lwa_exch_rate-pr_flag = abap_true.
```

Setting a session break-point in ABAP editor



The screenshot shows a code editor window in the SAP Integration Hub. The code is written in ABAP. A red box highlights the line of code at line 21:

```
17 *Validate if data is in the past  
18  
19  
20 *Loop data records of in coming proxy message  
21 LOOP AT lwa input-exchange rates-exch rate INTO lwa exch rate.  
22  
23 *Check if from date is in the past  
24 IF lwa_exch_rate-valid_from < sy-datum.  
25 lv_error_flag = abap_true.
```

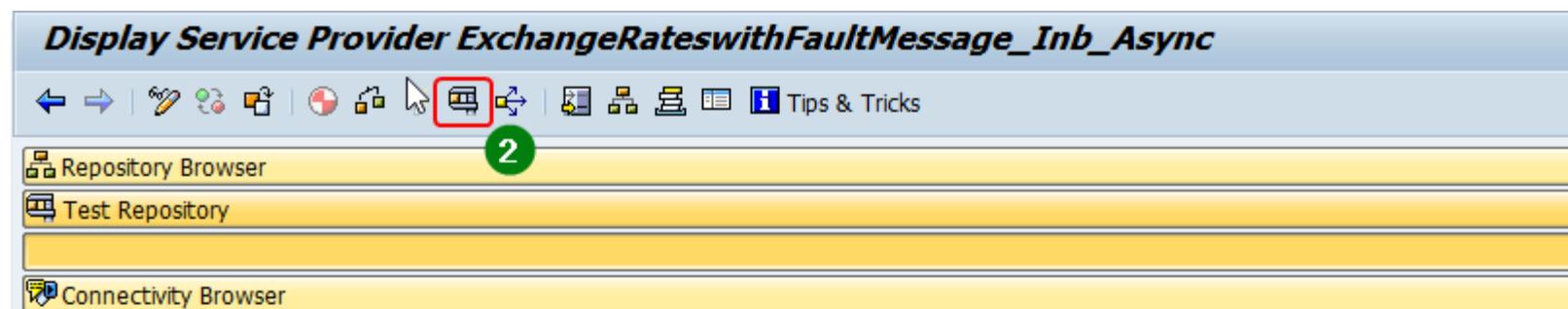
A session break-point is assigned at line 21, indicated by a small icon in the margin next to the line number.

Session break-point assigned in ABAP code

Step 3 – Select “Test Service Provider”

Go back to the main screen of SPROXY transaction after setting the break-point.

To simulate the inbound proxy click on the “Execute” button on the main menu. This will trigger the “Test Service Provider” function of SPROXY.



The screenshot shows the SAP Integration Hub interface. At the top left is the SAP Integration Hub logo. On the right are three icons: a magnifying glass, a search icon, and a close/cancel icon.

The main area displays two tables:

- Enterprise Service Repository (ESR) Table:**

Category	Description
ESR	Enterprise Service Repository
SWCs	<ul style="list-style-type: none"> INTERFACEDEMO INTERFACEDEMO, 1.0 of pocasts.com MAPPING MAPPING, 1.0 of integrationhub.com SENDERSYSTEM SENDERSYSTEM, 1.0 of integrationhub.com SOURCE_SYSTEM SOURCE_SYSTEM, 1.0 of integrationhub.com SWCV_APTOS SWCV_APTOS 1.0 of trg.com SWCV_AVERY SWCV_AVERY 1.0 of averydennison.com SWCV_BILLTRUST SWCV_BILLTRUST 1.0 of billtrust.com SWCV_MAPPING SWCV_MAPPING, 1.0 of mapping SWCV_THIRDPARTY_1 SWCV_THIRDPARTY_1, 1.0 of thirdparty-vendor-1
- Service Provider Configuration Table:**

<ul style="list-style-type: none"> urn:Target_System:ProductDetail urn:Target_System:DecodedString urn:Target_System:Base64EncodedString urn:Target_System:Base64OutputMessage urn:Target_System:ExchangeRates <ul style="list-style-type: none"> Object Types <ul style="list-style-type: none"> Data Types Fault Message Types Service Providers <ul style="list-style-type: none"> ExchangeRates_Inb_Async ExchangeRateswithFaultMessage_Inb_Async Message Types Objects 	
<ul style="list-style-type: none"> urn:Target_System:ExchangeRates <ul style="list-style-type: none"> ExchangeRates_Inb_Async ExchangeRateswithFaultMessage_Inb_Async 	urn:Target_System:ExchangeRates <ul style="list-style-type: none"> 1
<ul style="list-style-type: none"> ESR (local) External Backend 	Enterprise Service Repository (backend view) Created from external WSDL/Schema Modelled locally in the backend system

Execute Service Provider to Test/Debug ABAP proxy

Step 4 – Define XML Input Options and Debugging Options



Option “**Generate Request Template**” for **input** is a very useful option. This option automatically generates an input XML message based on the XSD or the input structure defined. For example, if you have an XML segment of type date in the input XML message, a default date will be set in the automatically generated sample XML message. Other XML segments of type string will set as String 1, String 2, etc.



The screenshot shows the SAP Integration Hub interface for configuring an ABAP proxy method. The 'Method Name' is set to 'EXCHANGE_RATESWITH_FAULT_MESSA'. Under the 'Input' section, the 'Generate Request Template' option is selected. Below it, there are two radio button options: 'xsi:nil for all Leaf Elements' and 'xsi:nil for nullable Leaf Elm.'. A checkbox for 'Deletion Indicator' is also present. The 'Debug method' section contains a checkbox for 'Don't catch application faults'. The bottom right corner features a toolbar with several icons, one of which is highlighted with a red border and a green circle containing the number '3'.

Debugging and XML input options of Proxy test in SPROXY

You can leave the debugging options blank as you have already set a session break-point manually in the ABAP code. But, using options "Debug Contractor" and "Debug Method" options, you can add **additional break-points** to the proxy runtime.

Step 5 – Define and Edit Input XML message

In the next screen, you can define the input XML message you would like to simulate. If you selected the option "Generate Request Template" in the previous step, you can simply change the values of the XML nodes as required.

To change the XML message generated by the system, click on "XML Editor" option in the header menu.



Request

```
- <n0:ExchangeRates xmlns:n0="urn:Target_System:ExchangeRates">
  - <EXCH_RATE>
    <RATE_TYPE>String 1</RATE_TYPE>
    <FROM_CURR>String 2</FROM_CURR>
    <TO_CURRENCY>String 3</TO_CURRENCY>
```



```
-----<FROM_FACTOR_V>8</FROM_FACTOR_V>
<TO_FACTOR_V>9</TO_FACTOR_V>
</EXCH_RATE>
- <EXCH_RATE>
  <RATE_TYPE>String 10</RATE_TYPE>
  <FROM_CURR>String 11</FROM_CURR>
  <TO_CURRENCY>String 12</TO_CURRENCY>
  <VALID_FROM>1999-01-24</VALID_FROM>
  <EXCH_RATE>13</EXCH_RATE>
  <FROM_FACTOR>14</FROM_FACTOR>
  <TO_FACTOR>15</TO_FACTOR>
  <EXCH_RATE_V>16</EXCH_RATE_V>
  <FROM_FACTOR_V>17</FROM_FACTOR_V>
  <TO_FACTOR_V>18</TO_FACTOR_V>
</EXCH_RATE>
</n0:ExchangeRates>
```

Privacy - Terms



upload XML messages from local files to the XML editor to define the XML input message.





The screenshot shows the SAP Integration Hub interface with the title "Request" at the top. A green circle with the number "1" is in the top right corner. The main area contains an XML document with numbered lines from 1 to 26. Lines 1 through 7 are highlighted with a red border. Lines 13 through 26 are also highlighted with a red border. The XML code is as follows:

```
<n0:ExchangeRates xmlns:n0="urn:Target_System:ExchangeRates">
  <EXCH_RATE>
    <RATE_TYPE>String 1</RATE_TYPE>
    <FROM_CURR>String 2</FROM_CURR>
    <TO_CURRENCY>String 3</TO_CURRENCY>
    <VALID_FROM>1999-01-24</VALID_FROM>
    <EXCH_RATE>4</EXCH_RATE>
  </EXCH_RATE>
  <EXCH_RATE>
    <RATE_TYPE>String 10</RATE_TYPE>
    <FROM_CURR>String 11</FROM_CURR>
    <TO_CURRENCY>String 12</TO_CURRENCY>
    <VALID_FROM>1999-01-24</VALID_FROM>
    <EXCH_RATE>13</EXCH_RATE>
    <FROM_FACTOR>14</FROM_FACTOR>
    <TO_FACTOR>15</TO_FACTOR>
    <EXCH_RATE_V>16</EXCH_RATE_V>
    <FROM_FACTOR_V>17</FROM_FACTOR_V>
    <TO_FACTOR_V>18</TO_FACTOR_V>
  </EXCH_RATE>
</n0:ExchangeRates>
```

XML editor screen of proxy test in SPROXY

Step 6 – Debug ABAP Code



The screenshot shows the SAP Integration Hub debugger interface. The top part displays the ABAP code for a method named ZII_EXCHANGE_RATESWITHFAULT. The code includes declarations for local work areas (lwa_bapi_input, lwa_input) and data structures (lt_return, lwa_standard_data, lwa_error_detail). The bottom part shows the execution stack (ABAP and Screen Stack), the current module (USER_COMMAND_0500), and the variables table.

```

5      lwa_bapi_input  TYPE bapi1093_0,
6          lwa_input      LIKE input.
7
8  DATA: lt_return          TYPE TABLE OF bapiret2,
9        lwa_standard_data  TYPE zexchange_fault_c
10       lwa_error_detail   TYPE zexchange_log_data
11
12
13
14
15
16
17  *Validate if data is in the past
18
19
20  *Loop data records of in coming proxy message
21  LOOP AT lwa_input-exchange_rates-exch_rate INTO
22
23  *Check if from date is in the past
24  IF lwa_exch_rate-valid_from < sy-datum.
25      lv_error_flag = abap_true.
26
27  *Append errors to exception table
28
29      lwa_standard_data-fault_text    = 'Error Occurred'
30      lwa_standard_data-fault_url     = 'https://sapintegrationhub.com/abap/proxy/debug-test-inbound-abap-proxy-sap-hana'
31

```

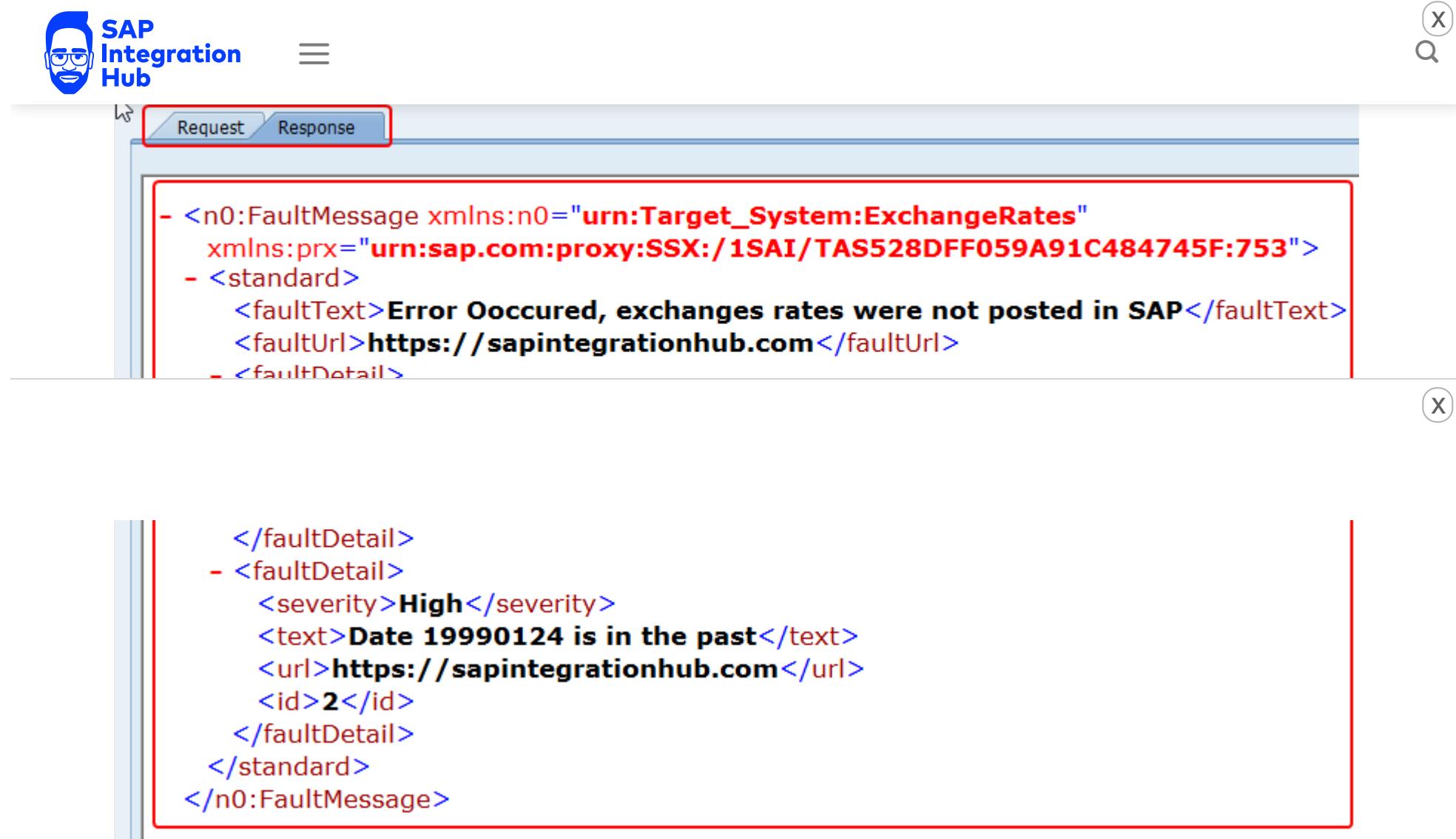
St...	Sta...	S...	Event Type	Event
14		METHOD	ZII_EXCHANGE_RATESW...	
13		METHOD	IF_PROXY_INBOUND_A...	
12		METHOD	EXECUTE	
11		METHOD	IF_PROXY_INBOUND_A...	
10		METHOD	INBOUND TEST	

Module (PAI)	User Command
PAI SCREEN	0500
FUNCTION	SPRX_SI_TEST
EVENT	START-OF-SELECTION

Variable	Value
LWA_EXCH_RATE-VALID...	19990124
SY-DATUM	20190712

Debug screen of SAP

When you complete the execution of the inbound proxy, XML response will be shown with the input message on the next screen. In this example, I have implemented [Fault Messages](#) so that application processing errors are returned back to the proxy runtime.



The screenshot shows the SAP Integration Hub interface. At the top, there's a navigation bar with the SAP Integration Hub logo, a search icon, and a refresh icon. Below the navigation bar, there are tabs for 'Request' and 'Response'. The 'Response' tab is selected, indicated by a red border around its tab. The main content area displays an XML fault message. A large red rectangular box highlights the entire XML structure. The XML code is as follows:

```
<n0:FaultMessage xmlns:n0="urn:Target_System:ExchangeRates"
  xmlns:prx="urn:sap.com:proxy:SSX:/1SAI/TAS528DFF059A91C484745F:753">
  - <standard>
    - <faultText>Error Occurred, exchanges rates were not posted in SAP</faultText>
    - <faultUrl>https://sapintegrationhub.com</faultUrl>
    - <faultDetail>
      - <severity>High</severity>
      - <text>Date 19990124 is in the past</text>
      - <url>https://sapintegrationhub.com</url>
      - <id>2</id>
    - </faultDetail>
  - </standard>
</n0:FaultMessage>
```

Response of the ABAP proxy with Fault messages

External Debugging method for Inbound ABAP Proxy



external systems.

But, before you can execute external debugging there are few pre-requisites that need to be full filled.

1. Proxy connectivity between PI/PO and ECC should be configured and working.
2. Proxy Object Generated and Interface ([iFlow](#)) Deployed

Prerequisite 1 – Proxy Connectivity Setup:

We need to have [Proxy connectivity](#) between SAP PI/PO system and SAP back-end system set up in order to develop working Proxy interfaces. There are multiple steps you need to complete to set up Proxy connectivity. You will need to coordinate with SAP BASIS team also to successfully complete the setup.

Prerequisite 2 – Proxy Object Generated and Interface Deployed:

This needs no explanation. You need to have a fully implemented [Inbound Proxy Interface](#) before you can start debugging or testing! From the implementation of PI/PO interface to the development of ABAP Proxy logic should be completed to complete the Proxy interface development.



Additionally, you can also refer how to configure acknowledgments in Inbound Proxy interfaces using [Fault Messages](#).

External Debugging Steps

Most of the steps to debug inbound proxies using external debugging method are the same as debugging using SPROXY. For example, finding the service interface, proxy class are the same for external debugging as SPROXY debugging. But, there are two very important distinctions between the two different methods.

To set up external debugging, additionally, you need to,

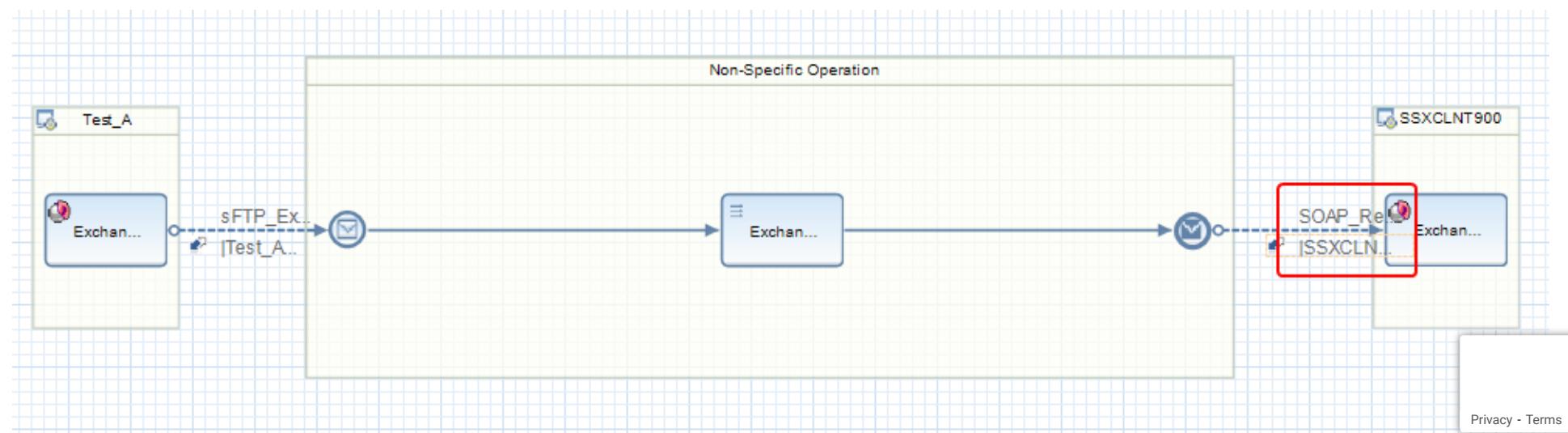
1. Find HTTP system user
2. Configure User-Specific Debugging Setting in SAP Back-end
3. Set an external debugging break-point instead of a session breakpoint

Step 1 – Find HTTP System Username

[Privacy](#) - [Terms](#)

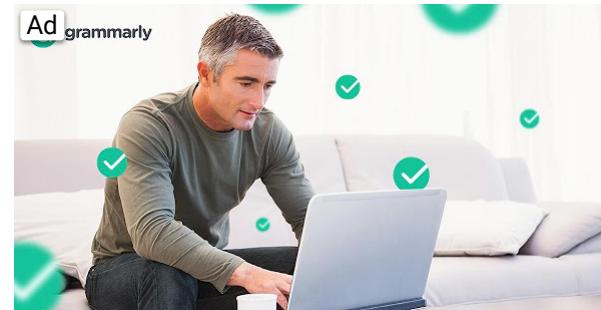
Similar to finding the service interface explained in the previous section of the article, you can find this using the iFlow or ICO. The trick is to find the **Receiver SOAP Communication Channel**.

Go to iFlow or ICO and find the Receiver SOAP Communication Channel.





Open Adapter Specific attributes tab of the SOAP Receiver Communication Channel to find the **HTTP Destination**.



Correct All Grammar Errors
Easily fix typos, grammatical mistakes, and other common issues before you hit Send
Grammarly [LEARN MORE](#)

General **Adapter-Specific** Modules Identifiers

General Advanced

Security Parameters

Select security profile

Connection Parameters

Addressing mode:* **HTTP destination**

HTTP destination:* **SSXCLNT900_HTTP**

Receiver Proxy (SOAP) Communication Channel Adapter Specific attributes

The next step is to find the **HTTP system user** used in the HTTP Destination. To find the user name, go to **NWA > Configuration Infrastructure > Destinations** and select the HTTP destination from the list.

Privacy - Terms



2

[Adobe Document Services Views](#)

Configure and manage the Adobe Document Services (credentials, trusted anchors, certificate revocation lists, license information, caches, and job profiles)

[Application Resources](#)

The Application Resources plug-in provides functionality to create, manage, and delete external resources such as JDBC drivers, JDBC data sources, resource adapters, and connection factories. The administrator can view all resources or choose a specific resource type and select a resource from the resources list. He or she can view its details and the details of all antecedent and dependent resources.

[Destinations Views](#)

Destinations are services that establish connections to other services. When using connections of this type, you need to specify the



Destinations in NWA

Then, go to the logon tab to find the **HTTP user**. In this example, the user name is "system". This user name is available in SAP back-end system. If you go to transaction SU01, you will be able to view the detail of this user.

This screenshot shows two windows from the SAP Integration Hub. The top window is titled 'Destination List' and contains buttons for 'Create...', 'Remove', 'Remove All', and 'Refresh'. It lists four destinations: 'Destination', 'FP_ICF_DATA_SSX' (highlighted in red), 'SLD_Client', and 'SLD_DataSupplier'. The bottom window is titled 'HTTP Destination SSXCLNT900_HTTP' and has tabs for 'Logon Data' and 'Basic Authentication'. It includes buttons for 'Edit' and 'Ping Destination'. Under 'Logon Data', it says 'Authentication: Basic (User ID and Password)'. Under 'Basic Authentication', the 'User Name:' field contains 'system' and the 'Password:' field contains a redacted password. A red box highlights the 'User Name:' field.

Destination List

Create... Remove Remove All Refresh

Destination

FP_ICF_DATA_SSX

SLD_Client

SLD_DataSupplier

Edit Ping Destination

HTTP Destination SSXCLNT900_HTTP

Logon Data

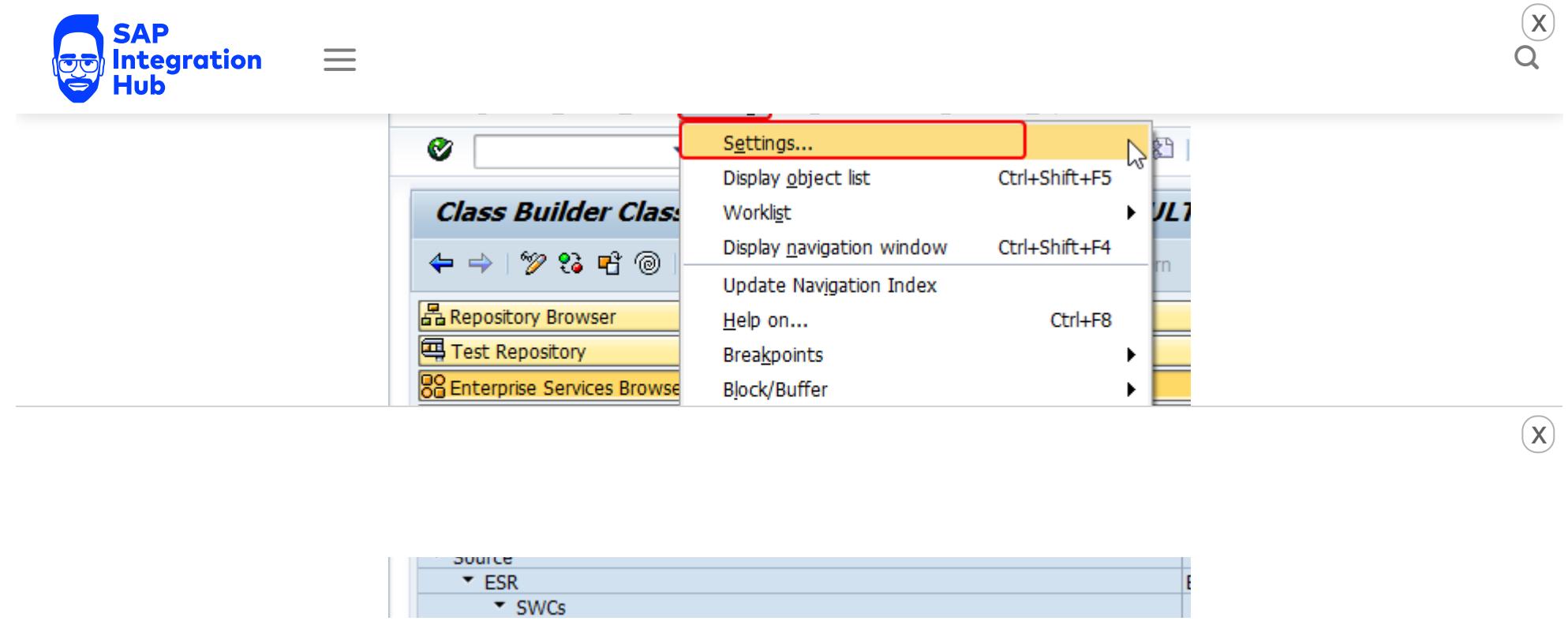
Authentication: Basic (User ID and Password)

User Name: system

Password: [REDACTED]

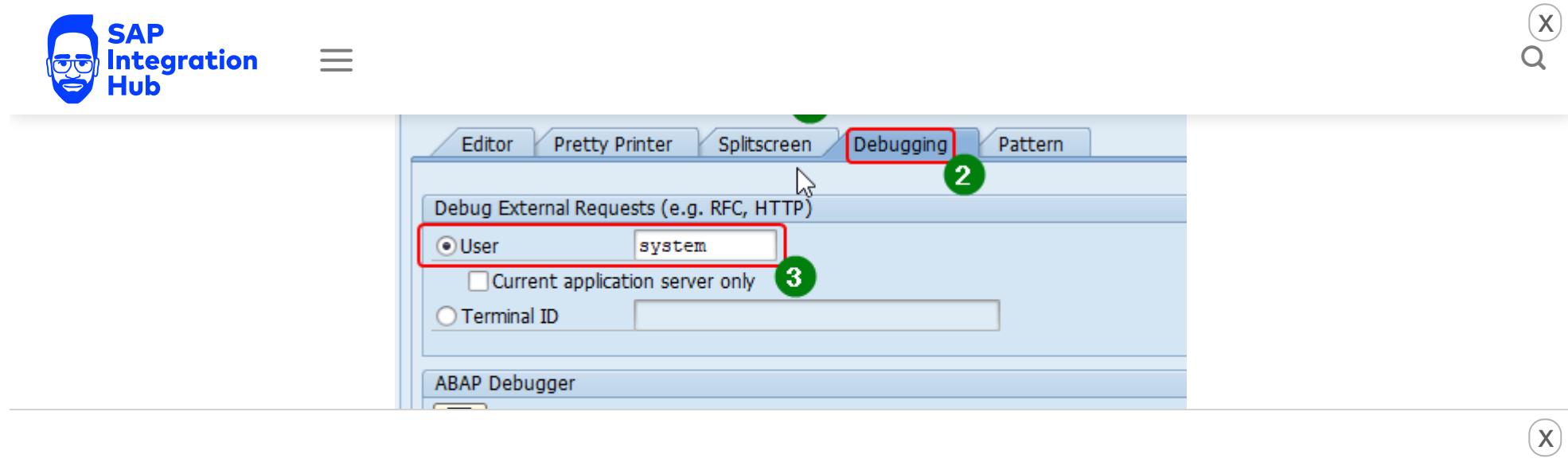
HTTP destination Logon Attributes

Step 2 – Configure User-Specific Debugging Setting in SAP Back-end



Access debug settings in SAP

Then set the user of Debug External Requests (RFC, HTTP) with the HTTP user we determined from the HTTP destination in PI/PO.



Debugging User-specific settings for external debugging (RFC, HTTP User)

Step 3 – Set External Debugging Point

Instead of a session debugging point, set an External Break-point in the Proxy ABAP code.



The screenshot shows the SAP ABAP editor interface. A context menu is open over line 25 of the code. The menu items are:

- More
- Set Session Breakpoint
- Delete Session Breakpoint
- Set External Breakpoint** (This option is highlighted with a yellow background and has a red border around it.)
- Delete External Breakpoint

The code itself is as follows:

```
24     IF lwa_exch_rate-valid_from < sy-datum.
25       lv_error_flag = abap_true.
26   ENDIF.
```

A tooltip or comment above the code reads: "date is in the past".

Settings the External Breakpoint in ABAP editor



```
Method ZII_EXCHANGE_RATESWITH_FAULT_M~EXCHANGE_RATESWITH_FAULT_MESSA active
13
14 *Assign proxy input message to a local variable
15 | lwa_input = input.
16
17
18 *Validate if data is in the past
19
```

Step 4 – Trigger Inbound Message from PI Test Tool

The next step is to trigger a message from the external system to SAP back-end system. Let do it via PI/PO test tool.

Using this method we decouple of dependency of having access to the message sender system or having to contact the message sender system owners. Maybe the message sender system is a third party system and you just want to test the message processing in SAP PI/PO and SAP back-end system.

This method skips the message processing pipeline step Sender Adapter message processing. This is a great way to test the PI/PO mapping, message routing, and SAP ABAP proxy application processing.

To send messages from PI Test tool, first, go to PI/PO home > Configuration and Monitoring home



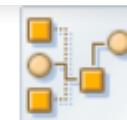
Enterprise Services Repository

Enterprise Services Builder | Web UI
Services Registry



Integration Directory

Integration Builder
Cloud Integration Content



System Landscape

System Landscape Directory



Configuration and Monitoring

Configuration and Monitoring Home
SAP NetWeaver Administrator



PI/PO Home and Configuration/Monitoring Home

Secondly, Navigate to **Testing > Send Test Message**.

Then, select the ICO from the drop-down and select the Quality of Service (QoS).

Monitoring Home

System: PO1

Monitoring Configuration and Administration **Testing**

Send Test Message
Simulate a message flow and verify that the SAP NetWeaver Process Integration runtime is functioning correctly by sending a test message to the Integration Engine or the Advanced Adapter Engine.

Test test message functionality in SAP PI/PO

Finally, enter the message Payload and hit Send! You can either copy paste the Payload or Upload it.

**Prefill Fields Based On:**Sender Agreement or Integrated Configuration (ICo): **[Test_A|ExchangeRates_Out_Async|urn:Source_System:ExchangeRates||]****Header Information**

Sender Party:

Sender Component:*******Test_A**

Receiver Party:



Receiver Component:

Interface:*******ExchangeRates_Out_Async**Interface Namespace:*******urn:Source_System:ExchangeRates**Quality of Service:*******Exactly Once****2****Payload** Enter Payload

```
<?xml version="1.0" encoding="UTF-8"?>
<ns0:ExchangeRates xmlns:ns0="urn:Source_System:ExchangeRates">
  <Rates>
    <FromCurrency>USD</FromCurrency>
    <ToCurrency>EUR</ToCurrency>
    <ExchangeRate>0.1</ExchangeRate>
    <ValidFromDate>01-03-2019</ValidFromDate>
  </Rates>
</ns0:ExchangeRates>
```

3 Upload File

Assign input XML message payload in send test message option

When the input message receives at the SAP back-end system, the system will automatically open a new session and processing will stop at the External debugging break-point. You can continue debugging the ABAP application login from here.



SAP
Integration
Hub

METHOD / ZII_EXCHANGE_RATESWITHFAULT_M~EXCHANGE_RA... SY-TABIX 0

Desktop 1 Desktop 2 Desktop 3 Standard Structures Tables Objects Detail Data Explorer Break./Watchpc

```
13
14 *Assign proxy input message to a local variable
15     lwa_input = input.
16
17
18 *Validate if data is in the past
19
20 *Loop data records of in coming proxy message
21     LOOP AT lwa_input-exchange_rates-exch_rate INTO lwa_exch_rate.
22
23         *Check if from date is in the past
24             IF lwa_exch_rate-valid_from < sy-datum.
25                 lv_error_flag = abap_true.
26
27         *Append errors to exception table
28
29             lwa_standard_data-fault_text    = 'Error Ooccured, exchanges rates
30             lwa_standard_data-fault_url    = 'https://sapintegrationhub.com'.
31
```

External debugging breakpoint activated in SAP

External Debugging Break-point Didn't trigger?

Check if inbound queues are activated using transaction SMQR. If the inbound qRFC queues are not registered, you will be able to find the queues in a "Scheduled" status in SMQ2.



Queue Information			
Number of Entries Displayed		1	
Number of Queues Displayed:		1	
C1.	Queue Name	Entries	
900	XBTR0002	1	

Inbound Queue of qRFC (Transaction SMQ2)

Additionally, you will be able to view these messages in a “Scheduled” status in SXMB_MONI.

To debug, you can activate the qRFC queue using the same transaction SMQ2.

qRFC Monitor (Inbound Queue)							
				Immediately	Without activation		
C1.	Queue Name	Entries	Status	Date 1	Time 1	NxtDate	NxtTim
900	XBTR0002	1	READY	12.07.2019	15:27:14	12.07.2019	15:27:14

Activate inbound queues in qRFC monitor (Transaction SMQ2)

I hope you are now able to understand the different inbound proxy testing and debugging methods. Also, steps on how to set up internal debugging using SPROXY and External HTTP debugging. If you have any questions on debugging and testing SAP ABAP Inbound Proxies, leave a comment below.



This entry was posted in [Proxy](#) and tagged [ABAP](#), [Break-Point](#), [External Debugging](#), [HTTP](#), [HTTP Destination](#), [ICO](#), [iFlow](#), [Implementing Proxy Class](#), [Inbound Proxy](#), [NWA](#), [NWDS](#), [PI Test Tool](#), [Proxy Connectivity](#), [qRFC](#), [SMQ2](#), [SMQR](#), [SOAP](#), [SOAP Receiver Adapter](#), [sproxy](#), [SXMB_MONI](#), [XML Editor](#).



ISURU FERNANDO

Hi, I am Isuru Fernando, Senior SAP Integration Consultant with 10 years of SAP full-cycle implementation and support project experience. From the early days, I had a passion for coding, software development, and even tech-related. I started my carrier as an ABAP developer and soon found my love for system integration when learned SAP XI 3.0 in 2008. Playing a variety of roles from an offshore technical consultant (ABAP, PI/PO, BW,

Privacy - Terms



and Distribution (SD), Material Management, Retail, Customer Relationship Management (CRM), and Finance and Controlling (FICO). Through this blog, I want to share my expertise in SAP technical areas such as SAP ABAP, PI/PO, AIF, and Basis. I also want to provide a platform for others with similar ambitions who would like to share their SAP technical expertise with the world!

[What is SAP PI \(PO\) – Ultimate Guide](#)

[How to Import Standard ESR Content to PI/PO](#)

SIGN UP TODAY!

Sign up to receive our monthly newsletter and special deals!

Your Name (Required)

Your Email (Required)



I accept Newsletters & Blog updates.

[SIGN UP](#)



[Privacy - Terms](#)



BECOME AN AUTHOR

Sign up as a contributing author to write your own articles!

[REGISTER](#)

TECH GADGETS I USE EVERY DAY

These are some of the tech gadgets I use every day. If you make a purchase through these links I will earn a small comission at absolutely no extra cost to you.



Bose
QuietComfort...
\$299.00 Prime

[Shop now](#)

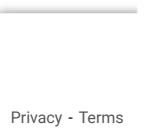


2019 Apple
MacBook Pro...
\$2,285.00 Prime

[Shop now](#)



(Renewed)
Apple iPhone 11
Pro
\$629.99





Nikon D3500
24.2MP DSLR...

Shop now

3 THOUGHTS ON “DEBUG AND TEST INBOUND ABAP PROXY IN SAP HANA (HOW TO GUIDE)”

mahi says:

Thank you for the excellent article. I enjoyed reading your blog!!

Privacy - Terms



Isuru Fernando says:

Thank you Mahi!!

AUGUST 9, 2019 AT 12:02 PM

REPLY



Richard says:

Is parasoft 2 use

MARCH 2, 2020 AT 3:48 PM

REPLY

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Privacy - Terms

[Website](#)[POST COMMENT](#)

1

[report this ad](#)

REGRESSION TEST AUTOMATION

[Privacy - Terms](#)



Automate SAP PO migration testing

TRY FOR FREE

Use promo code
"SAPIntegrationHub"
to extend the free trial period
of Int4 IFTT by one week

TAG CLOUD





The ezoic logo, which is a green circle with a white 'e' and 'o'.

report this ad

RECENT COMMENTS

Jakku on [Change iDoc Status by Standard Program](#)

pavani on [XML Transformation Example with XSLT_TOOL](#)

Jamaic on [iDoc Collection and Package Size – S4 HANA and PI/PO Configuration](#)

Nila on [Outbound IDoc Configuration with Output Determination in SAP – Techno-functional Guide](#)

Rituparna Roy on [Proxy Outbound Interface Example SAP to PI File Receiver](#)

LATEST POSTS

09 Jul [Integrating VIES VAT Validation with SAP S4 HANA](#)

Privacy - Terms

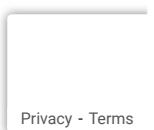
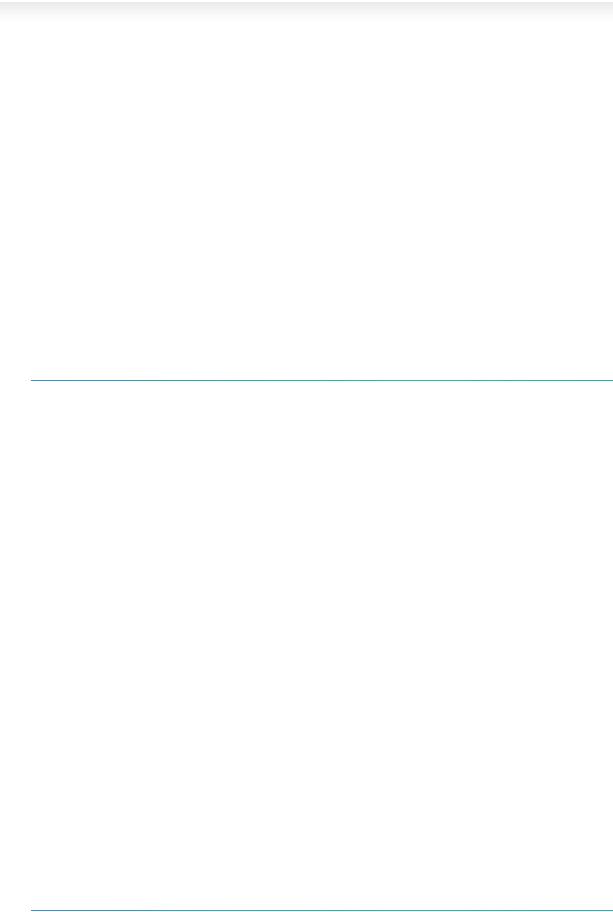
21
Mar[POS Integration with SAP S4 HANA and CAR](#)

2 Comments

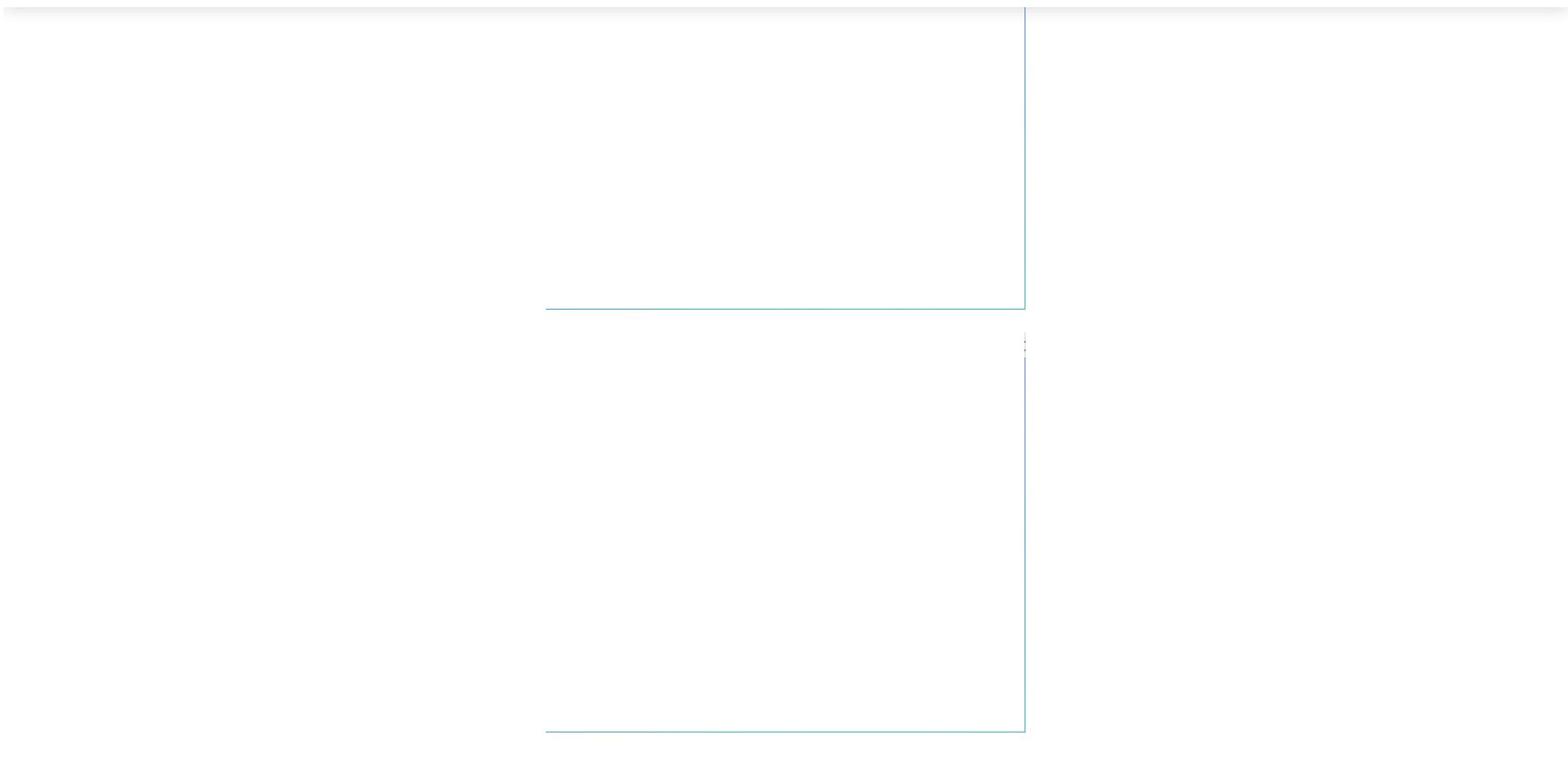
06
Mar[Ultimate Guide to SAP PI/PO to CPI Migration](#)

CATEGORIES

[ABAP](#)[AIF](#)[BASIS](#)[CPI](#)[How To](#)[Integration Architecture](#)[Integration Scenarios](#)[PI/PO](#)



Privacy - Terms



SITE LINKS

[Home](#)

[Privacy](#) - [Terms](#)

[Disclaimer](#)[Privacy Policy](#)[Contributing Author Registration](#)

SIGN UP TODAY!

Sign up to receive our monthly newsletter and special deals!

Your Name (Required)

Your Email (Required)



I accept Newsletters & Blog updates.

[SIGN UP](#)



[Privacy - Terms](#)

