SAP Integration Hub

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 it's 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.

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 ECC and SAP PI/PO 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.

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

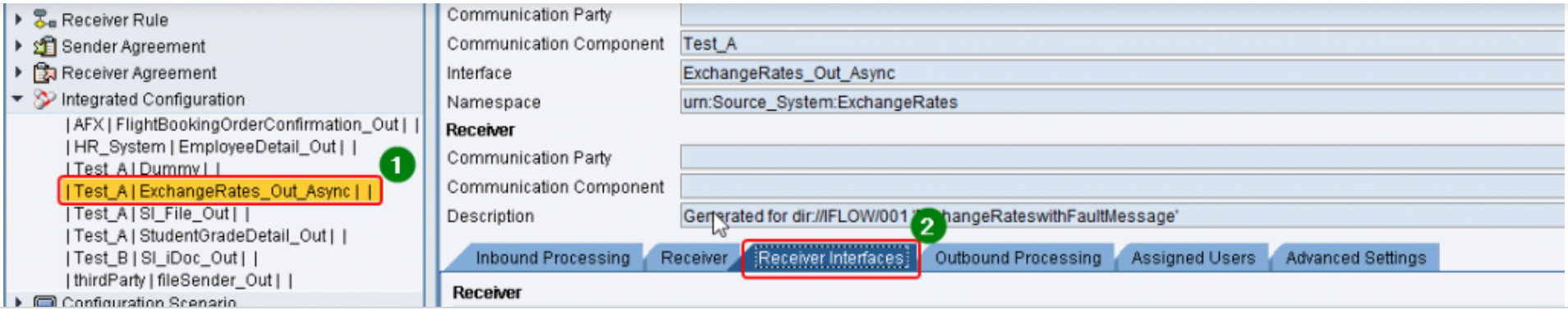
## Step 1 – Find the Inbound Service Interface and Namespace

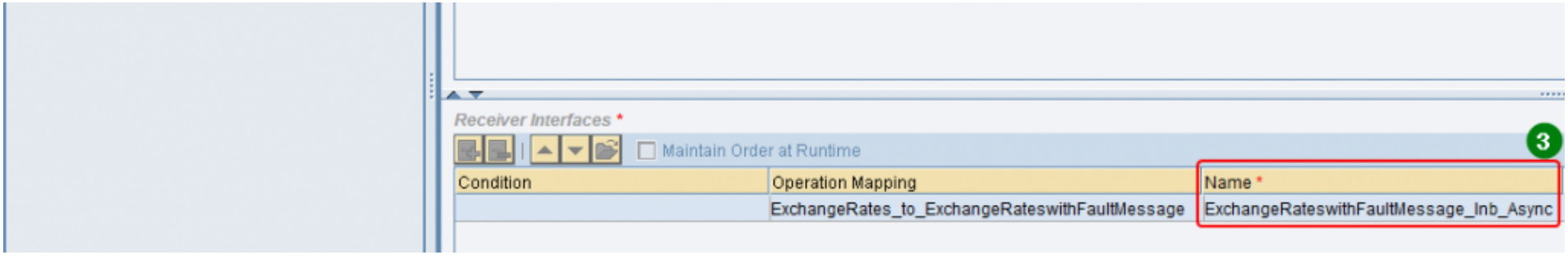
- Option 1 – Find the Inbound Service Interface via the Integrated Configuration Object (ICO):
- Option 2 – Find Interface using iFlow in NWDS:

### 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.



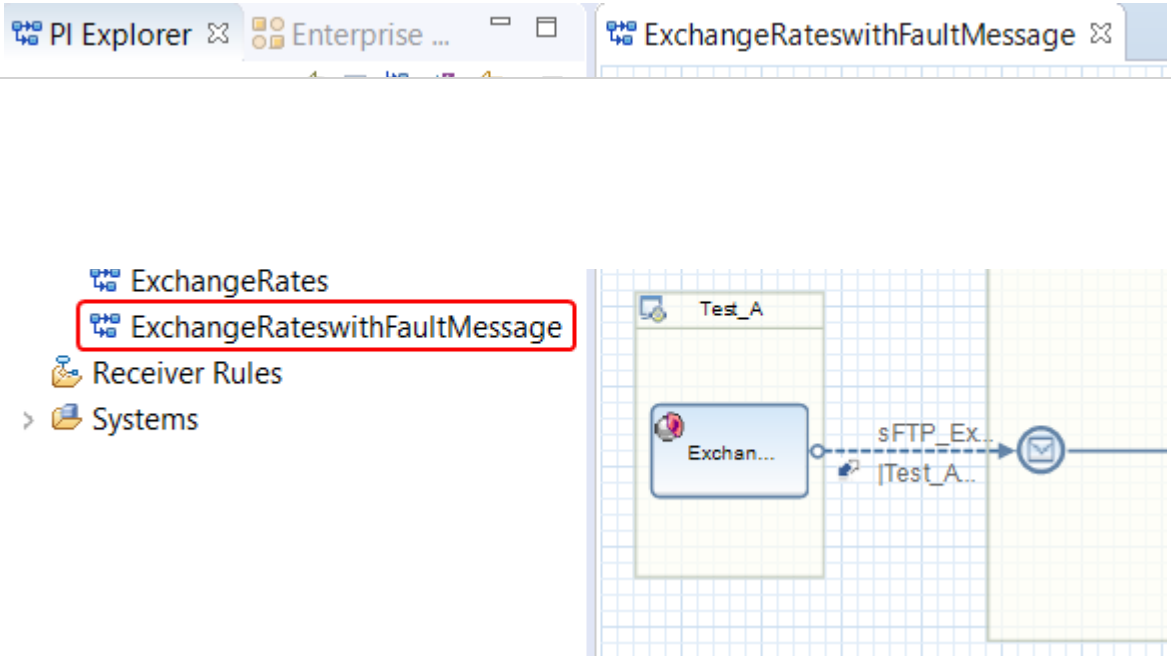




Receiver Interface in ICO in ESR

## Option 2 – Find Interface using iFlow in NWDS:

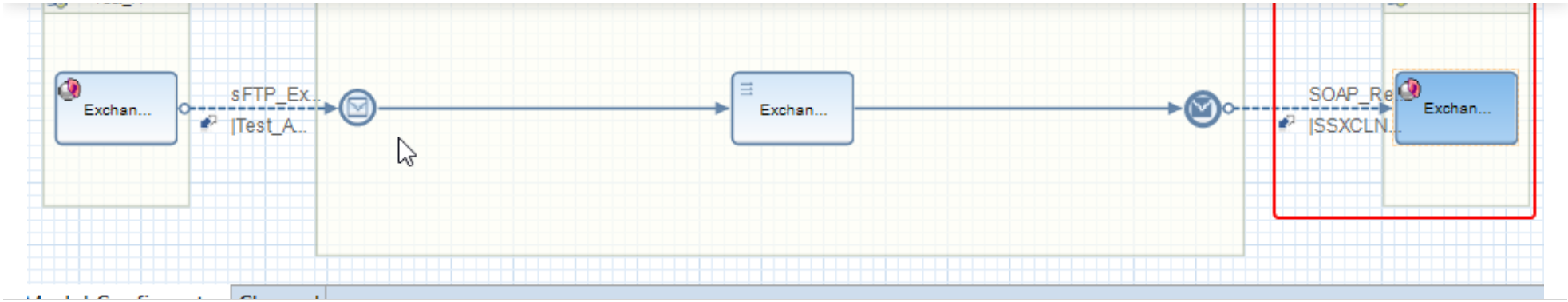
If you have implemented the interface using iFlow, we can also find the Interface detail directly via the iFlow.

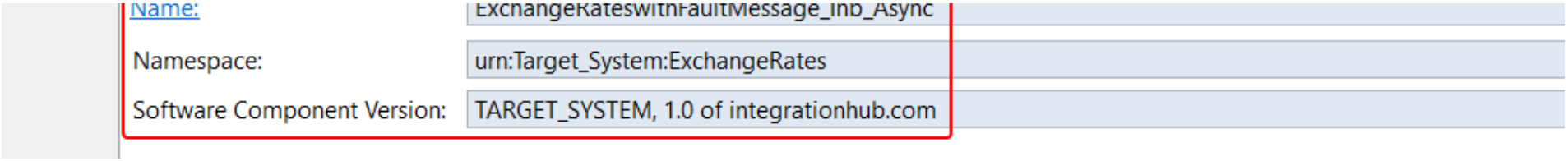
Open NWDS and navigate to your iFlow under the Integration Flow navigation menu.




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.



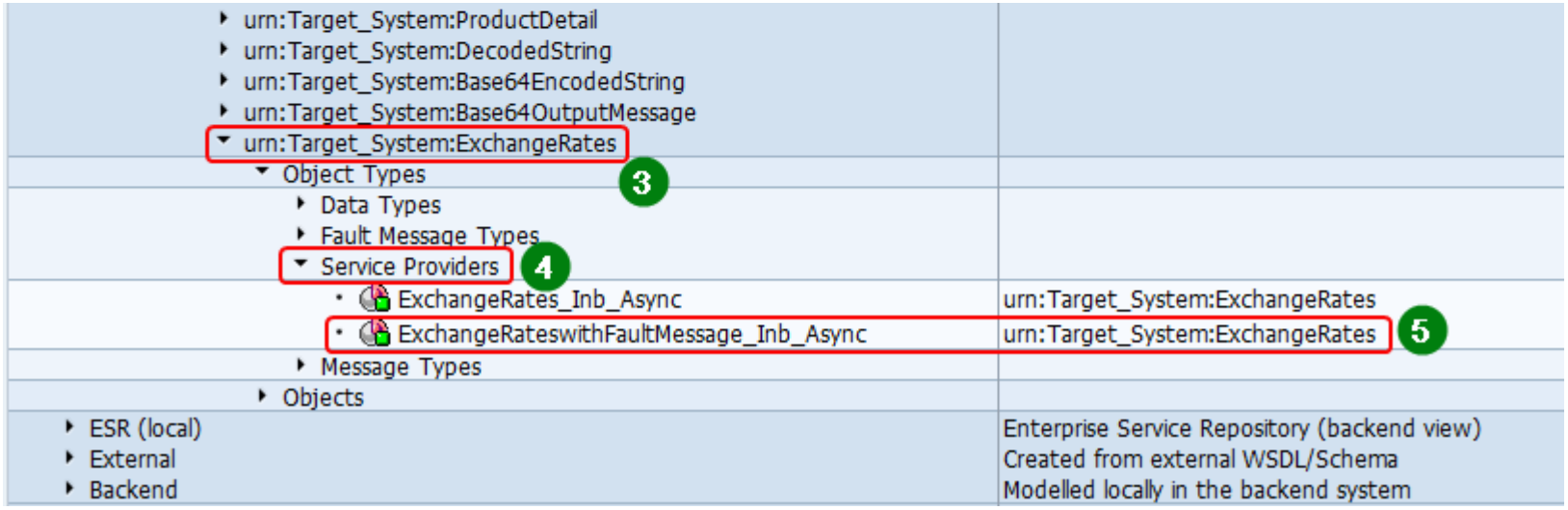




Interface detail (name, namespace and SWCV) in iFlow

## Step 2 – Determine the ABAP Proxy Class Using SPROXY Transaction

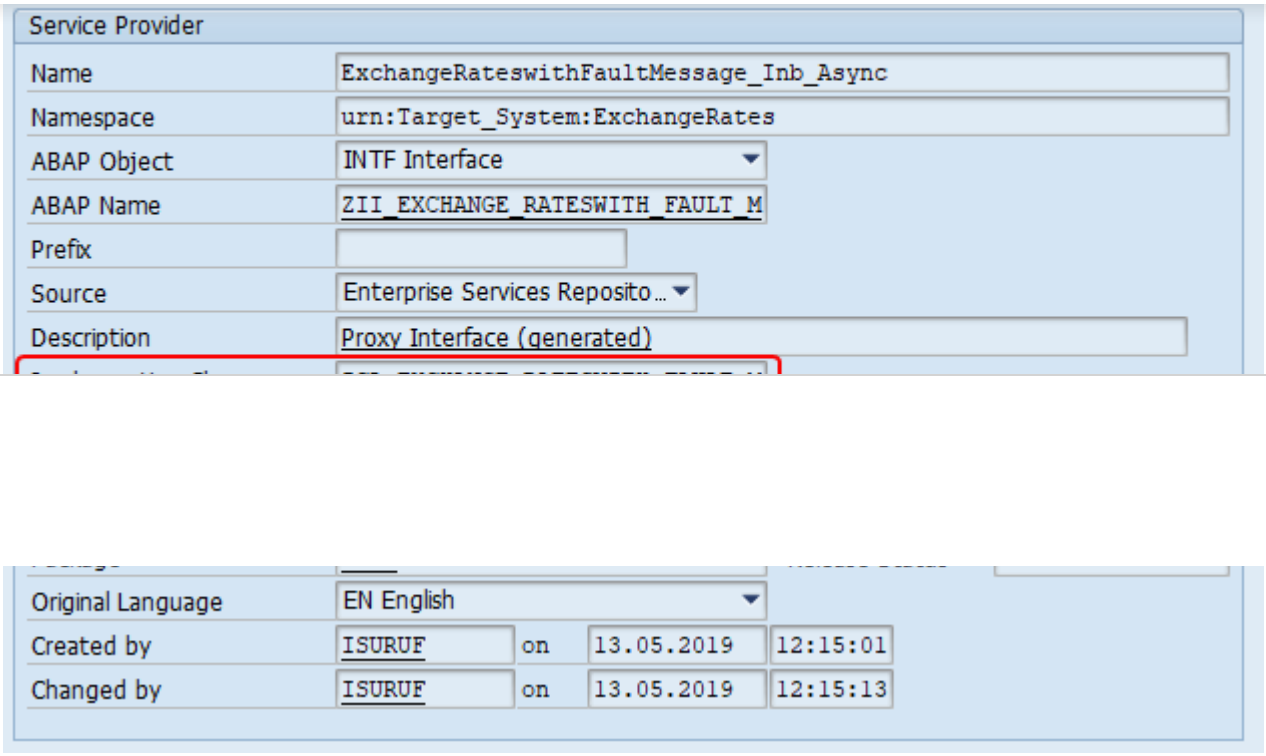
Now that you have identified the Inbound Service Interface details such as name, SWCV, and Namespace, go to transaction SPROXY in SAP back-end system and navigate to the interface using the navigation tree.

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


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.



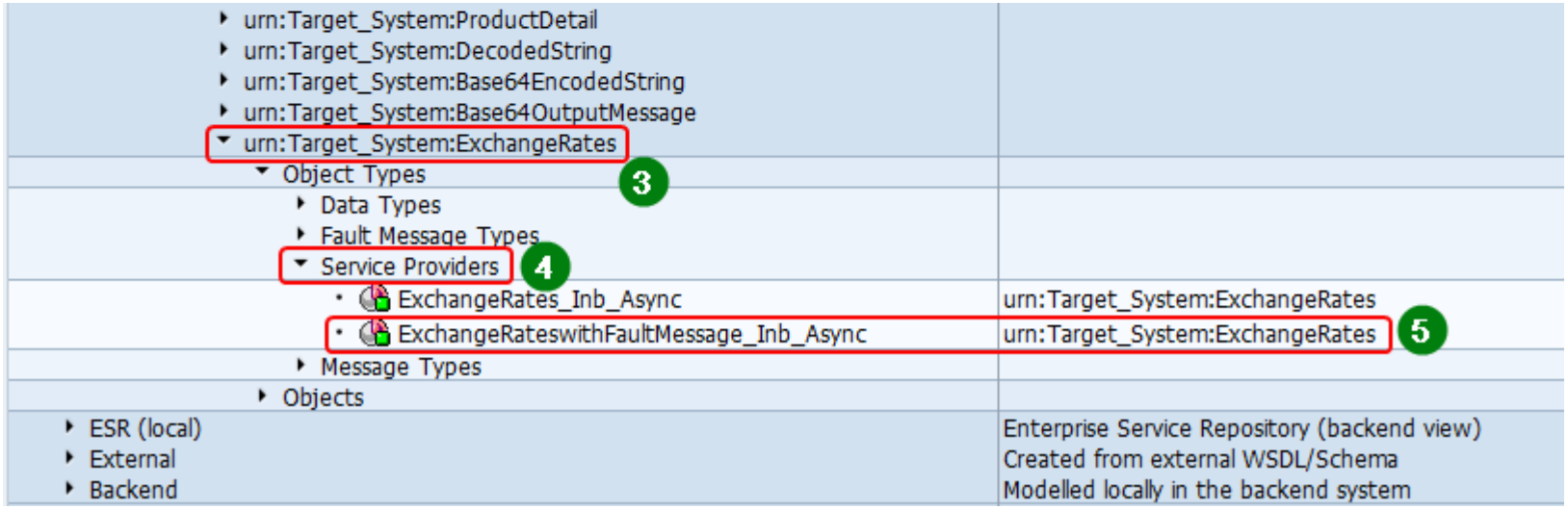

Implementing Class of SAP ABAP Proxy

# 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

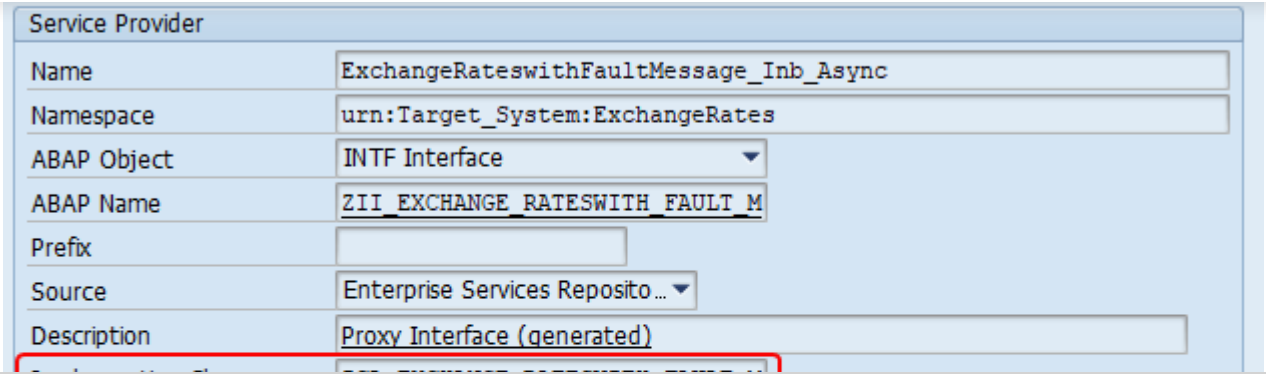
As we discussed in the previous chapter, go the inbound Service Interface in the transaction SPROXY. You can use the navigation tree menu of SPROXY transaction to find your Service Interface and Proxy Class.

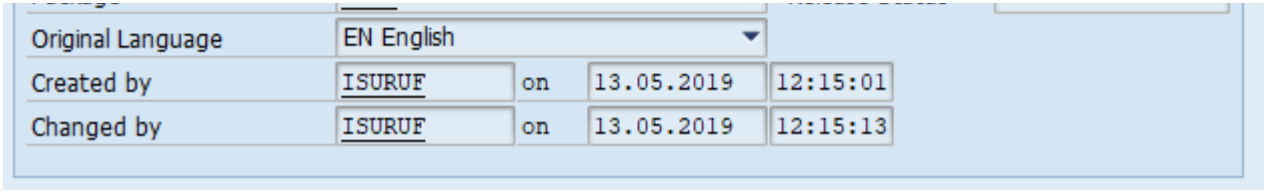




Navigation menu of SPROXY in SAP back-end system

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

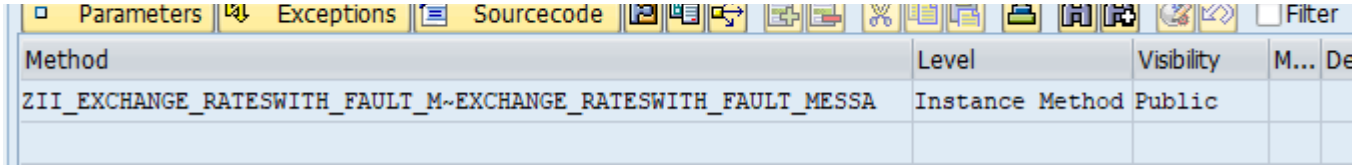
Go to ABAP proxy method by double-clicking on the “Implementing class”.



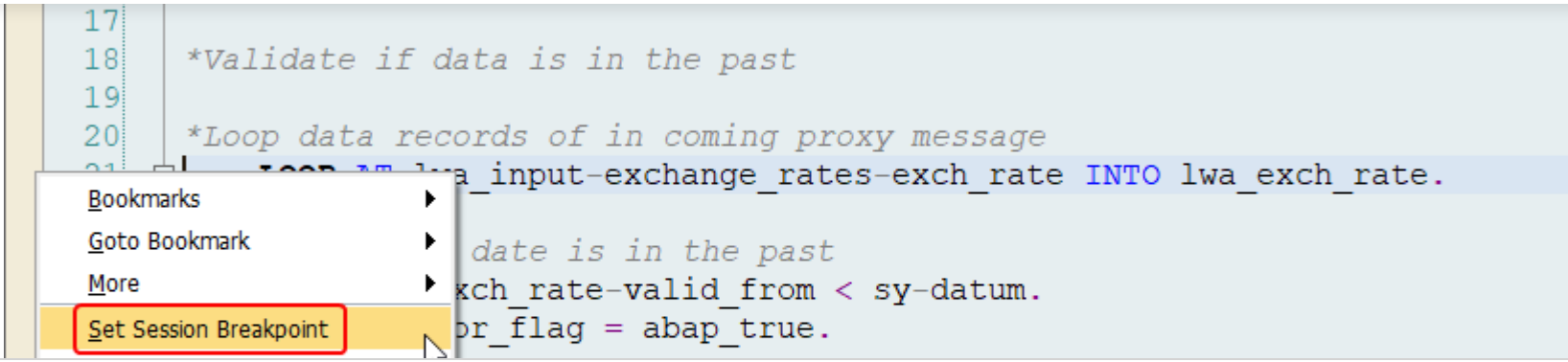

Implementing class of Service Provider in SAP back-end

Then navigate to the instance method of the Proxy class.




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.




Setting a session break-point in ABAP editor

```
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.
```

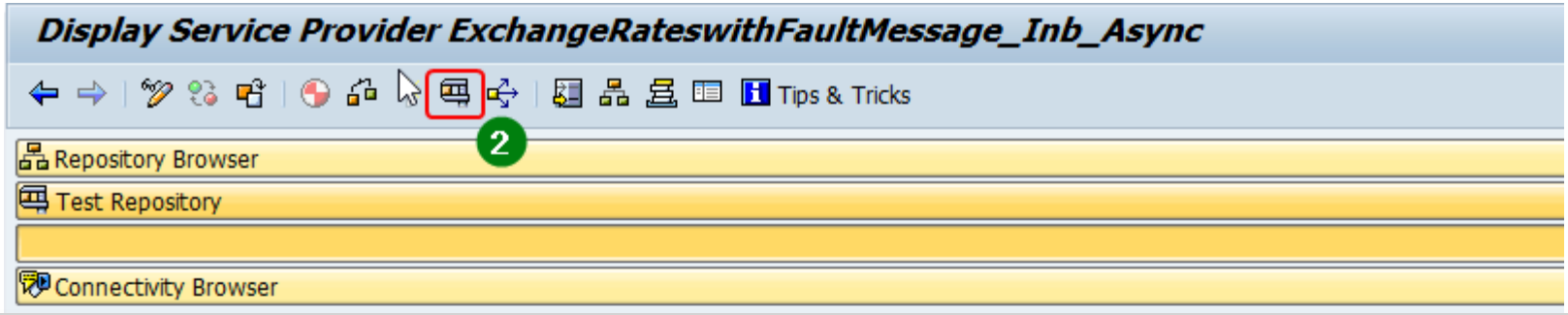
```
31
```

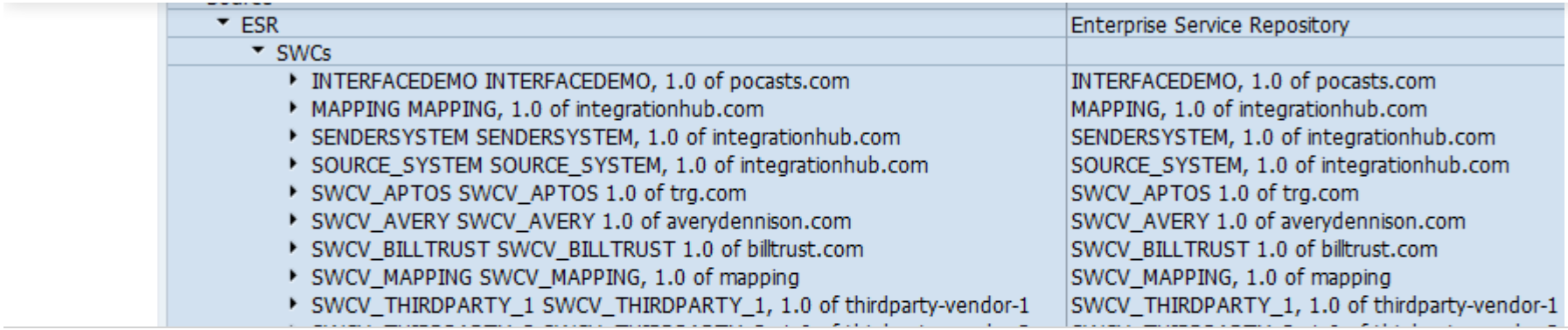
Session break-point assigned in ABAP code

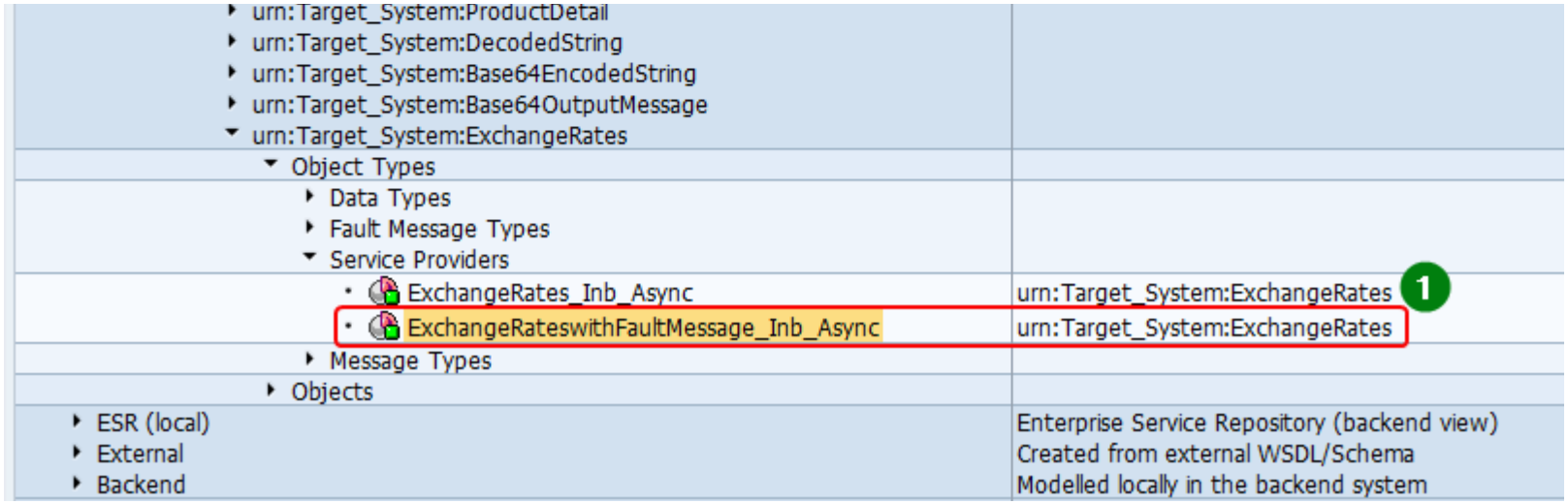
## Step 3 – Select "Test Service Provider"

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.



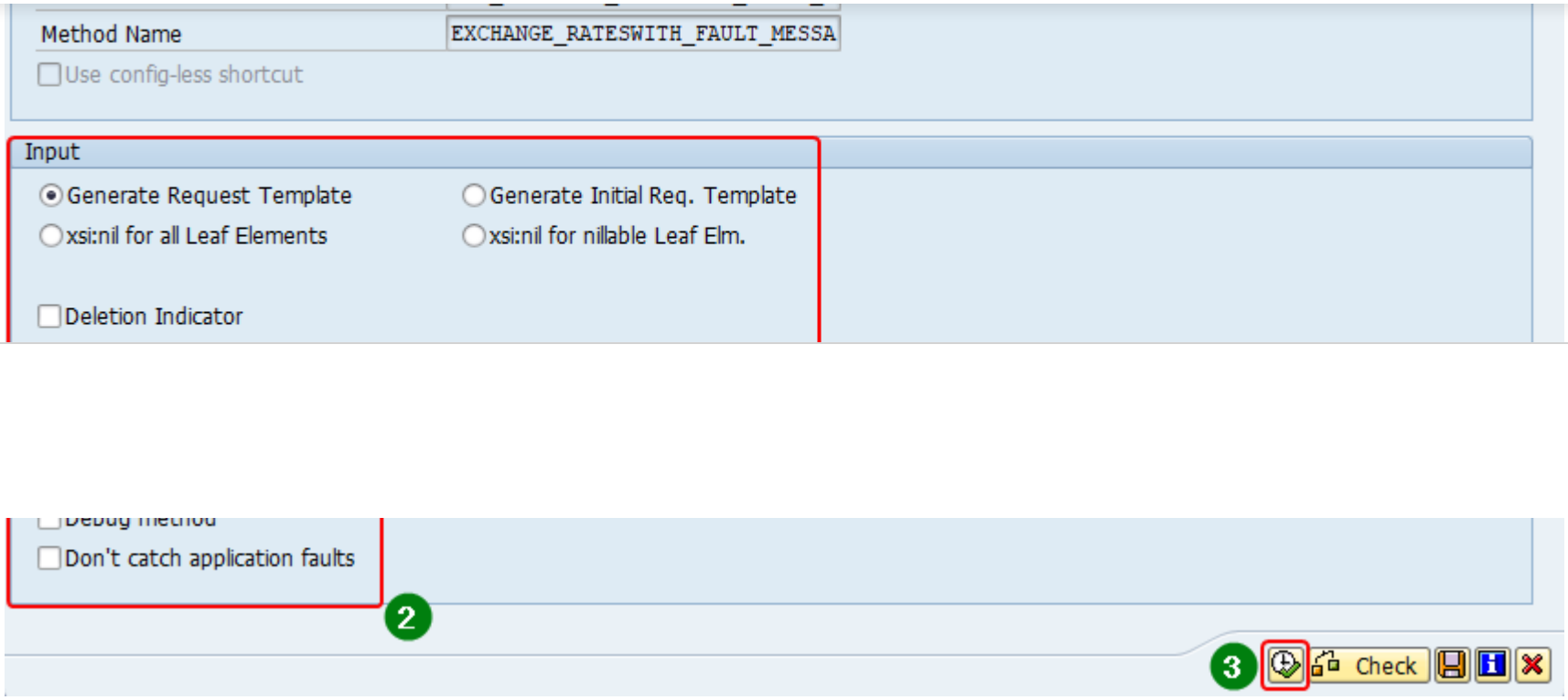



Execute Service Provider to Test/Debug ABAP proxy

## 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.



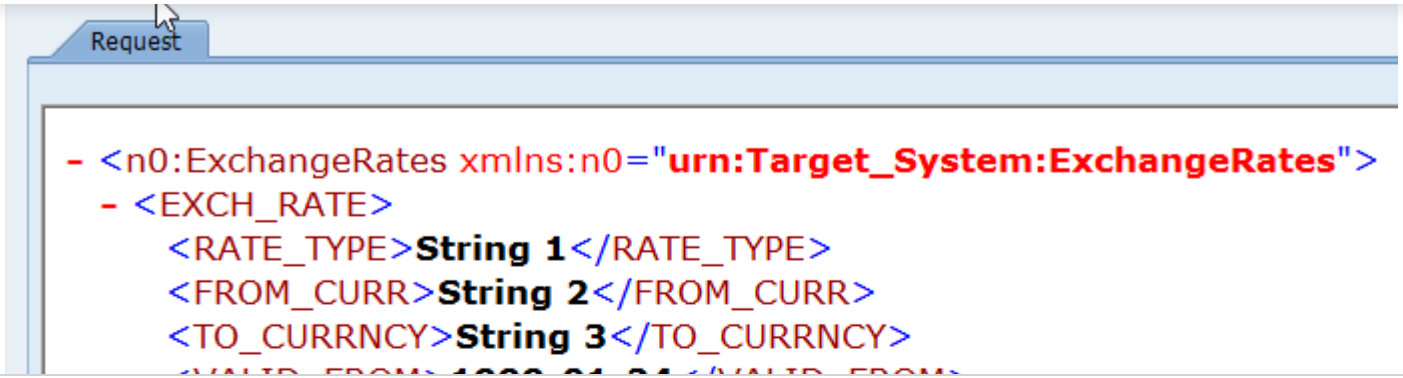

Debugging and XML input options of Proxy test in SPROXY

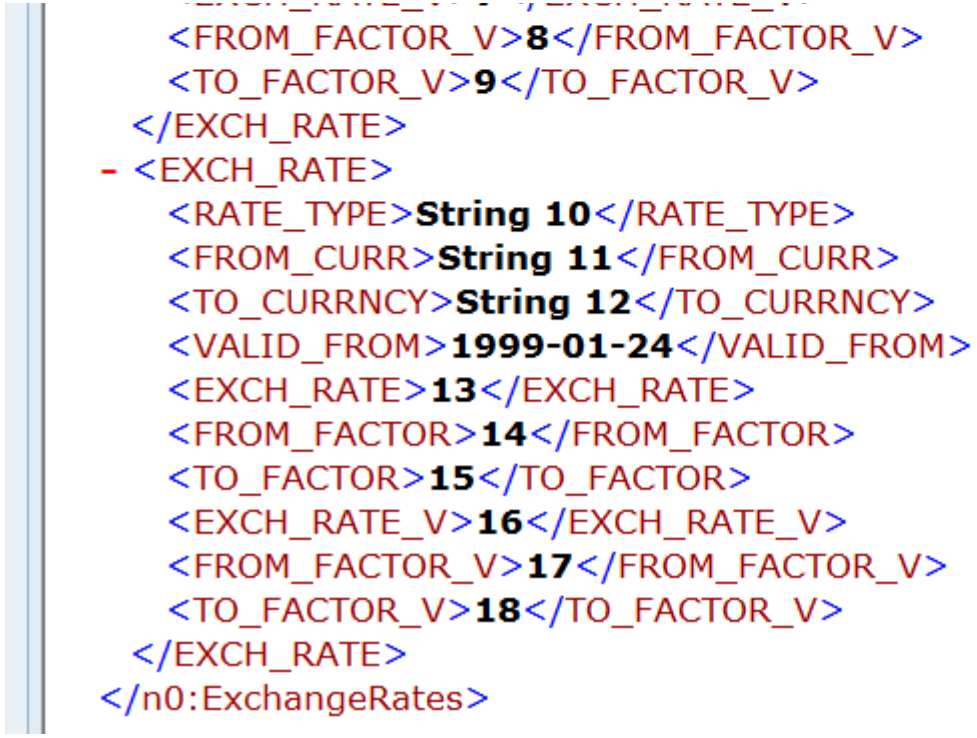
You can leave the debugging options blank as you have already set a session break-point manually in the ABAP code. But, using options "**Debug Contractor**" and "**Debug Method**" options, you can add **additional break-points** to the proxy runtime.

## 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.

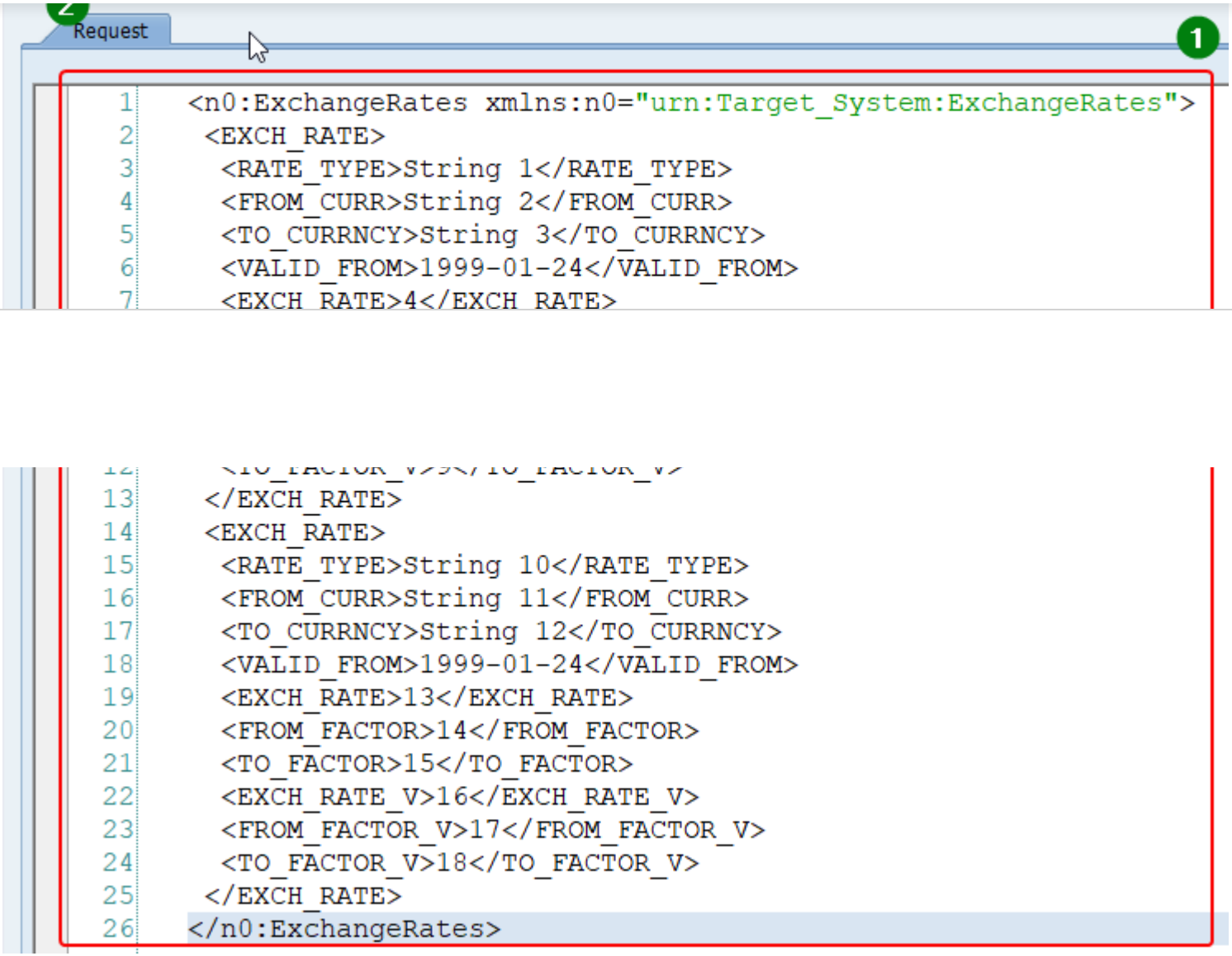
To **change the XML message** generated by the system, click on "**XML Editor**" option in the header menu.



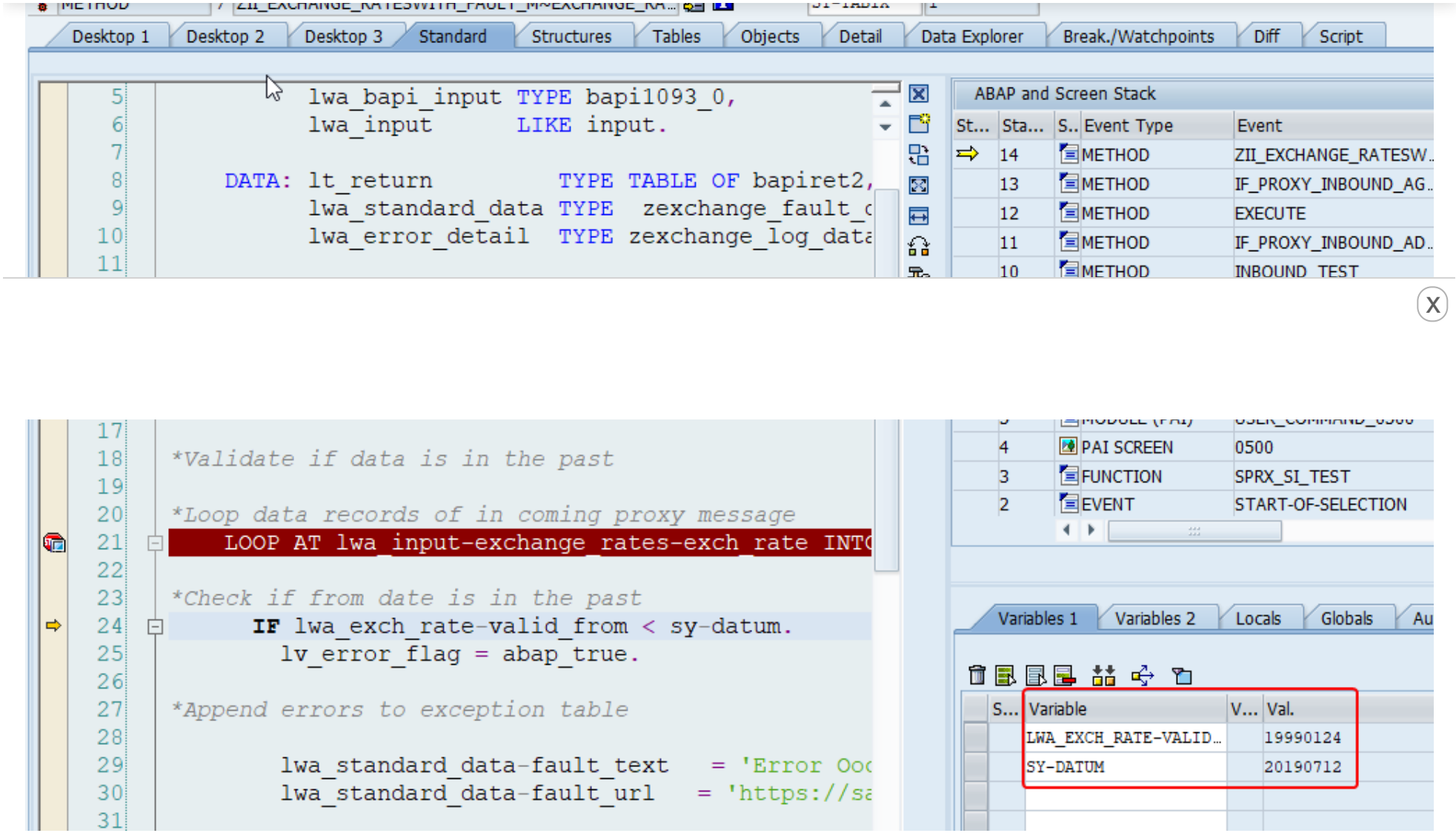

Edit XML input of the ABAP proxy inbound message

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



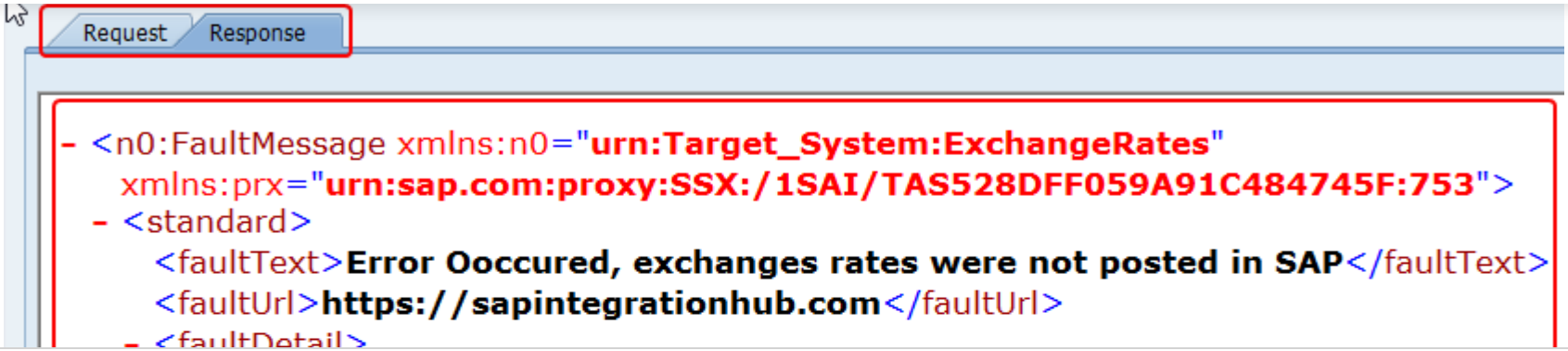

XML editor screen of proxy test in SPROXY

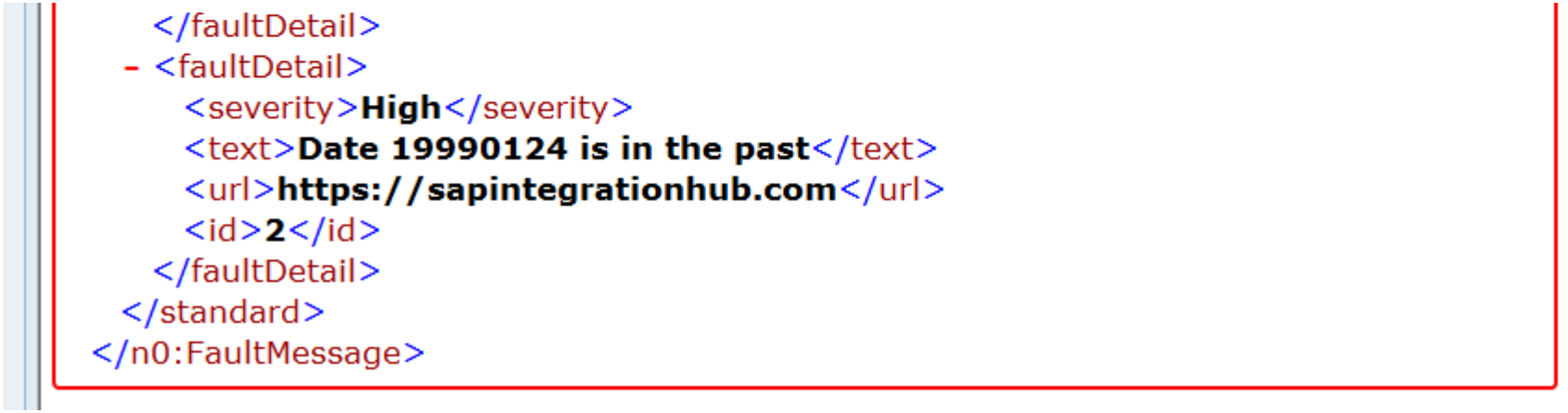
## Step 6 – Debug ABAP Code




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 sc
In this example, I have implemented Fault Messages so that application processing errors are returned back to the proxy runtime.




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 o testing! From the implementation of PI/PO interface to the development of ABAP Proxy logic should be completed to compl 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.

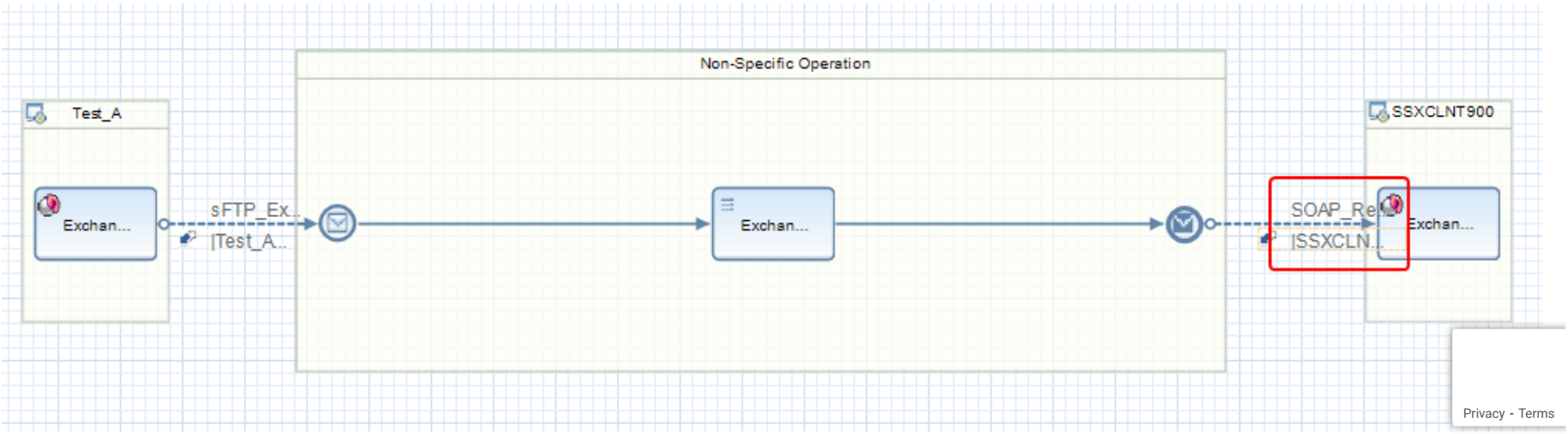
To set up external debugging, additionally, you need to,

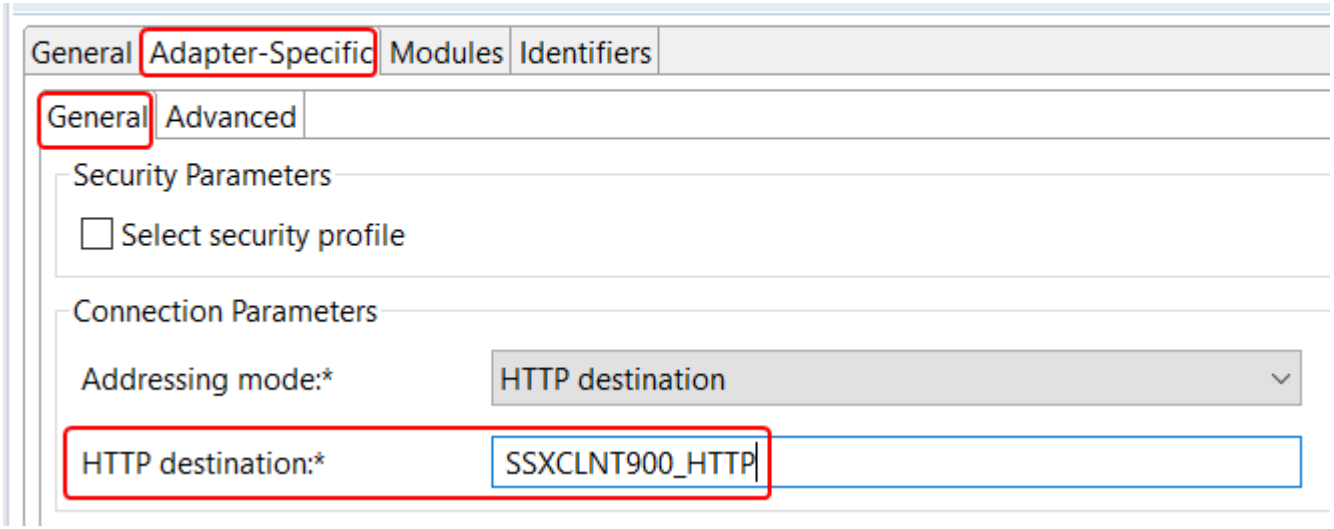
1. Find HTTP system user
2. Configure User-Specific Debugging Setting in SAP Back-end
3. Set an external debugging break-point instead of a session breakpoint

### Step 1 – Find HTTP System Username

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**.

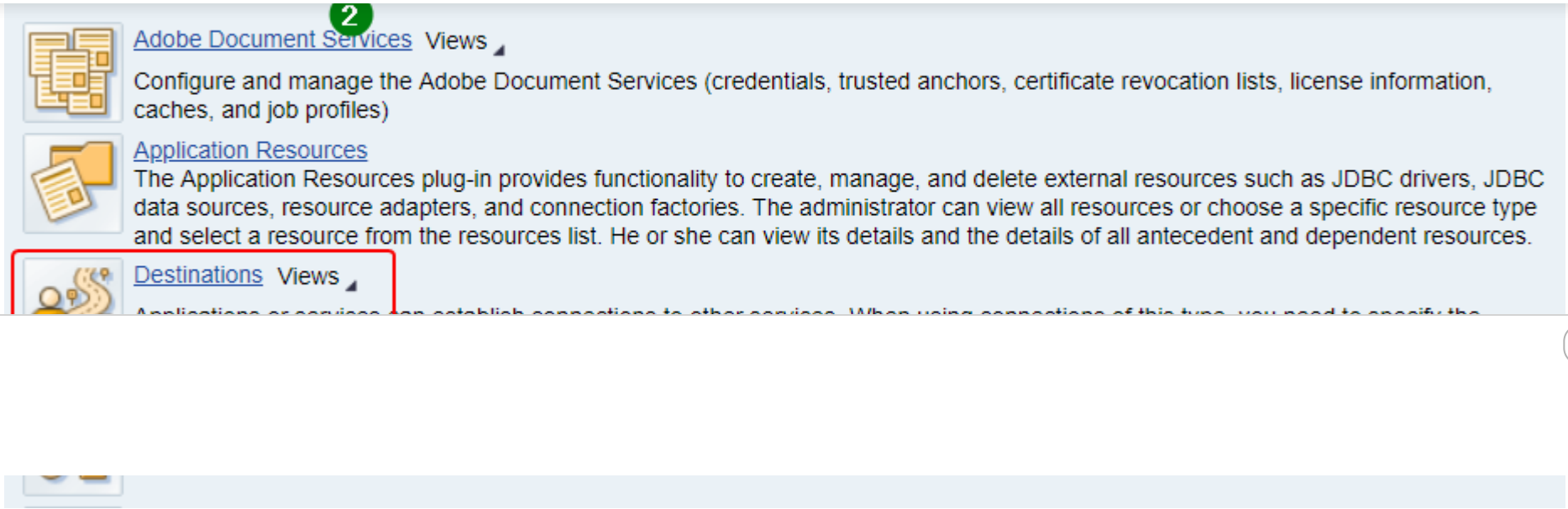
Go to iFlow or ICO and find the Receiver SOAP Communication Channel.





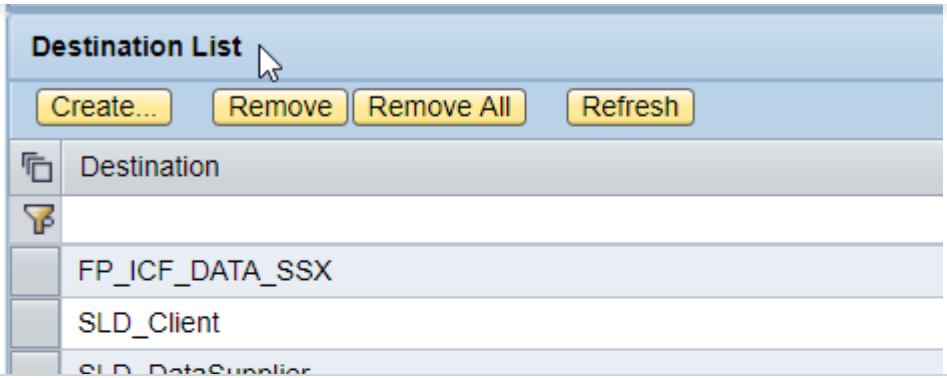

Receiver Proxy (SOAP) Communication Channel Adapter Specific attributes

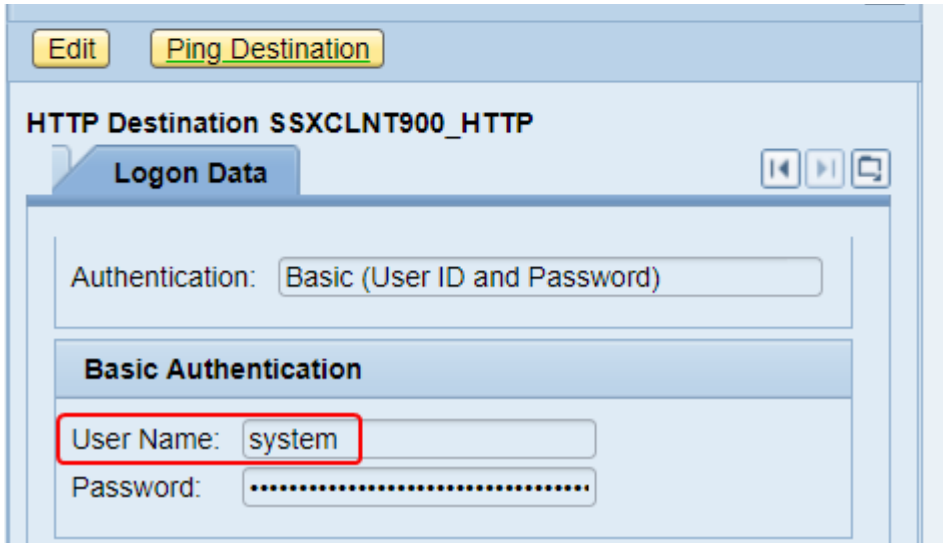
The next step is to find the **HTTP system user** used in the HTTP Destination. To find the user name, go to **NWA > Configurati Infrastructure > Destinations** and select the HTTP destination from the list.




Destinations in NWA

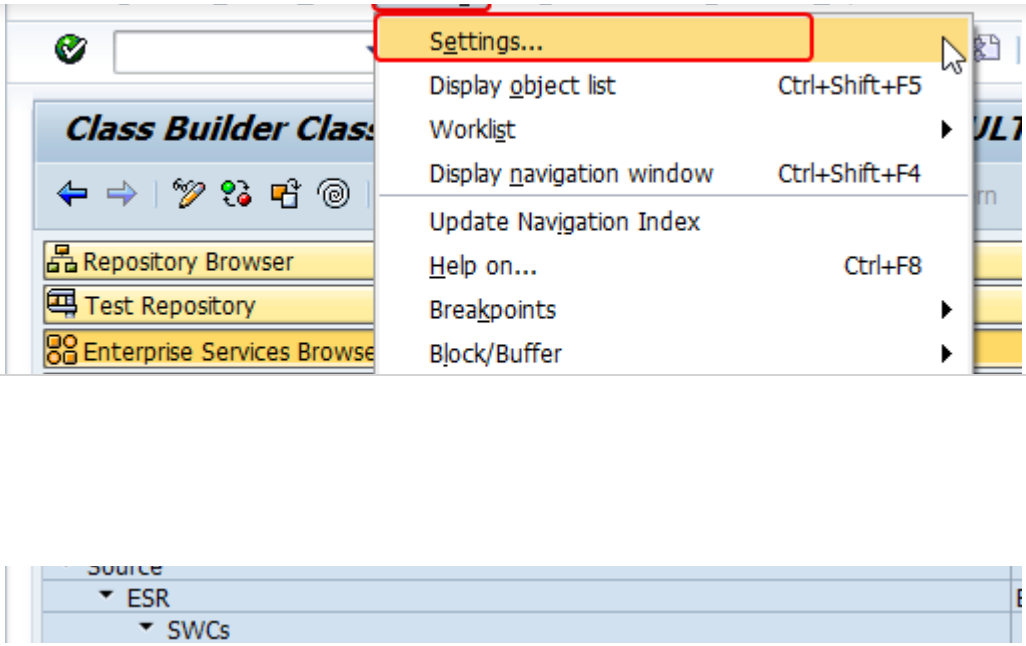
Then, go to the logon tab to find the **HTTP user**. In this example, the user name is “system”. This user name is available in SAP back-end system. If you go to transaction SU01, you will be able to view the detail of this user.



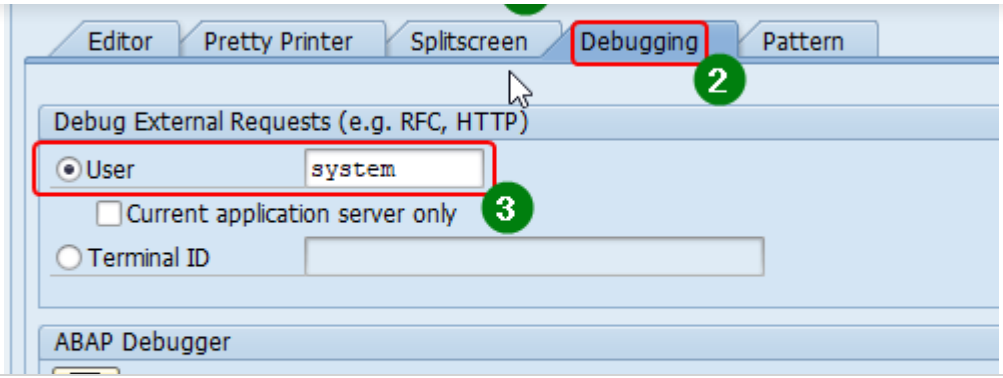

HTTP destination Logon Attributes

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




Access debug settings in SAP

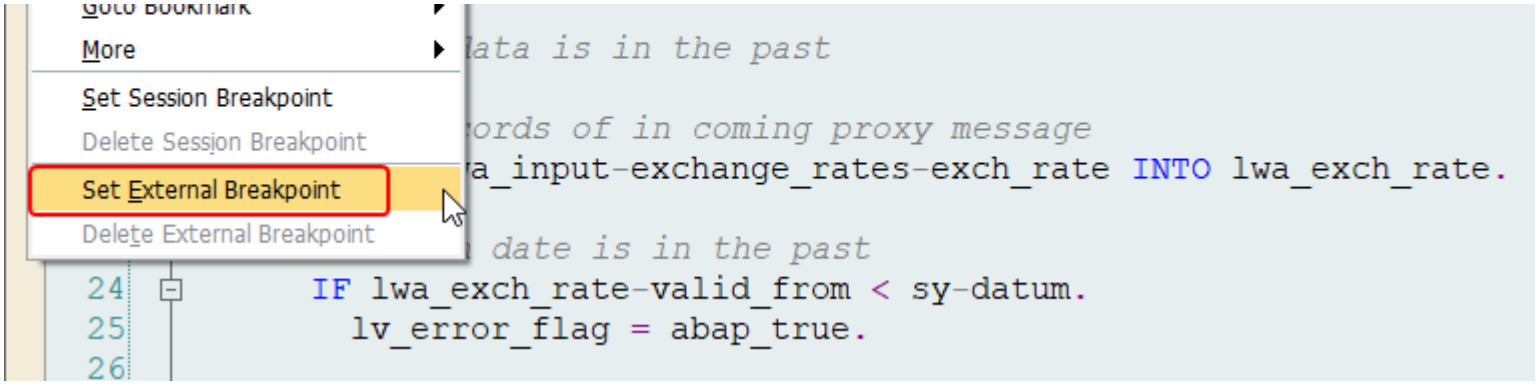
Then set the user of Debug External Requests (RFC, HTTP) with the HTTP user we determined from the HTTP destination in PI/PO.



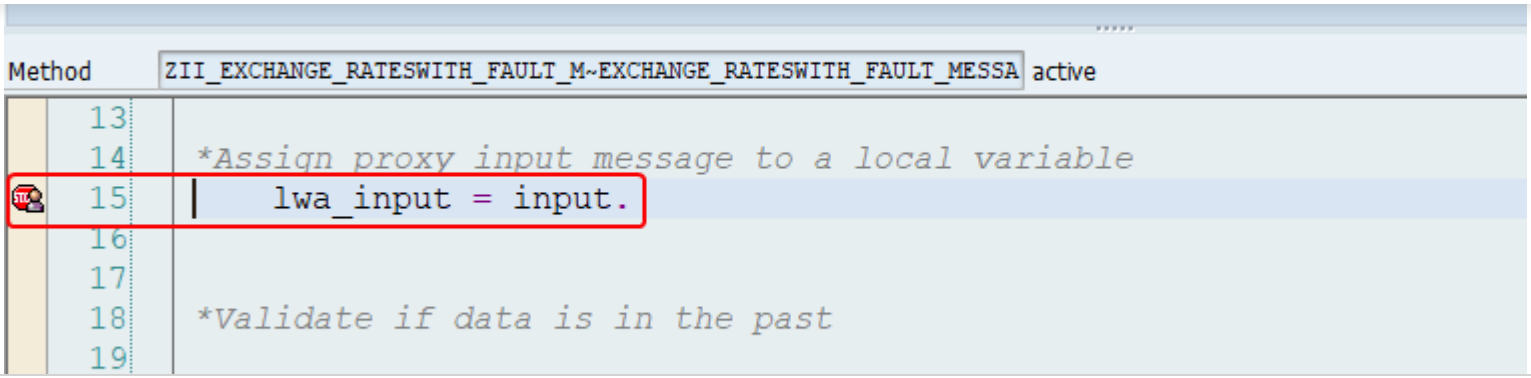

Debugging User-specific settings for external debugging (RFC, HTTP User)

## Step 3 – Set External Debugging Point

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




Settings the External Breakpoint in ABAP editor




## 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.

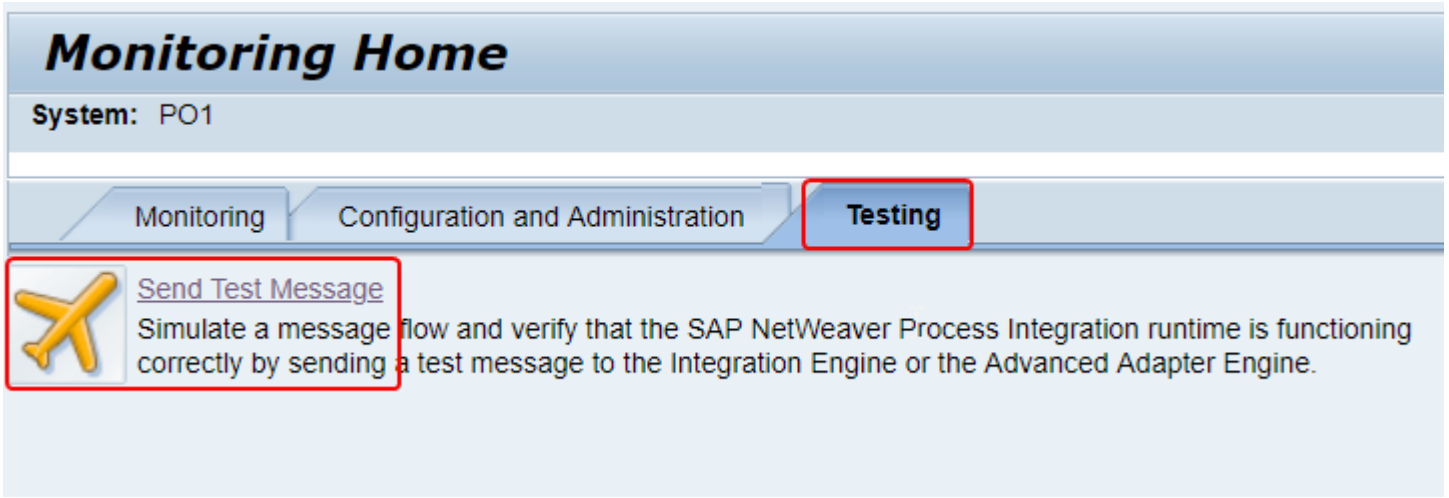
This method skips the message processing pipeline step Sender Adapter message processing. This is a great way to test the PI/PO mapping, message routing, and SAP ABAP proxy application processing.

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


PI/PO Home and Configuration/Monitoring Home

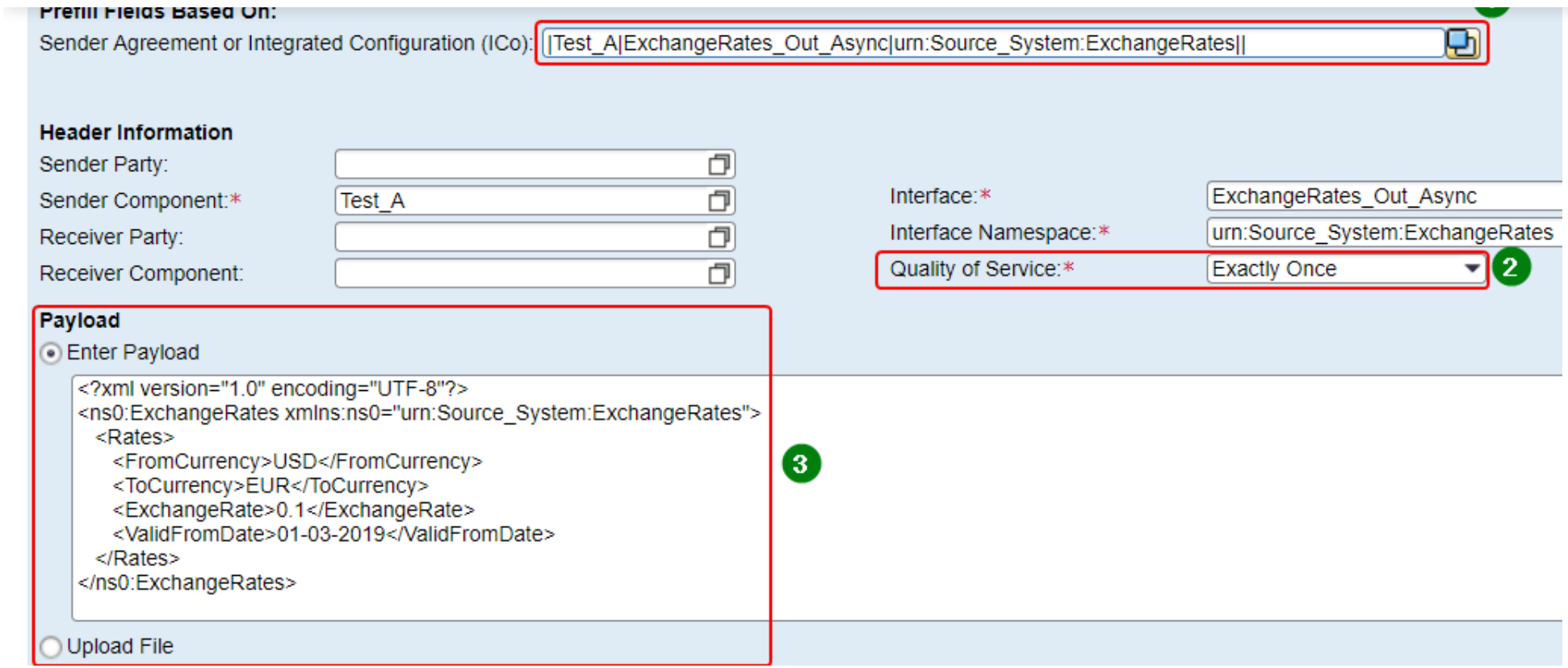
Secondly, Navigate to **Testing > Send Test Message**.

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



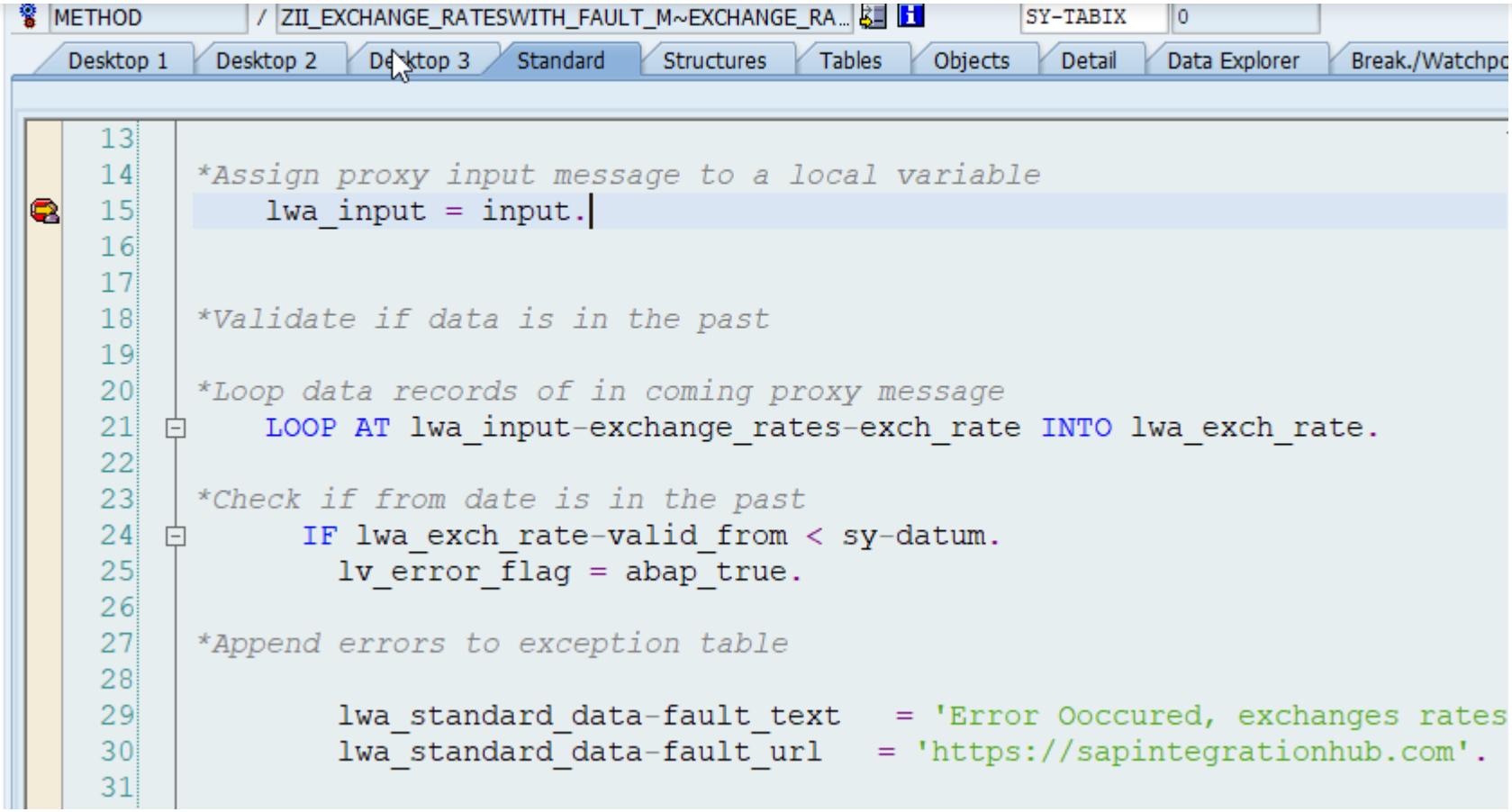

Test test message functionality in SAP PI/PO

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




Assign input XML message payload in send test message option

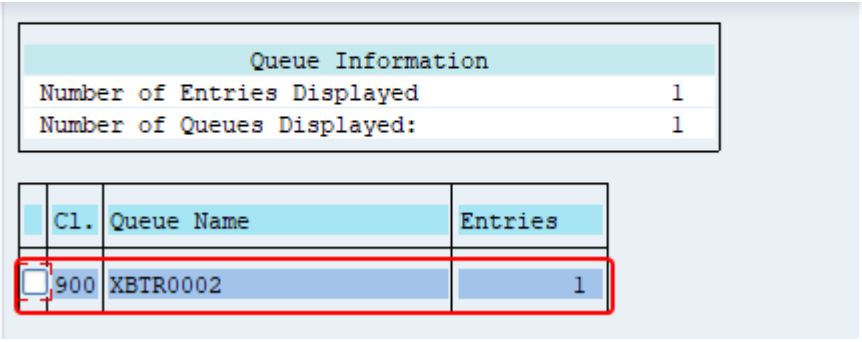
When the input message receives at the SAP back-end system, the system will automatically open a new session and processing will stop at the External debugging break-point. You can continue debugging the ABAP application login from here.




External debugging breakpoint activated in SAP

## External Debugging Break-point Didn't trigger?

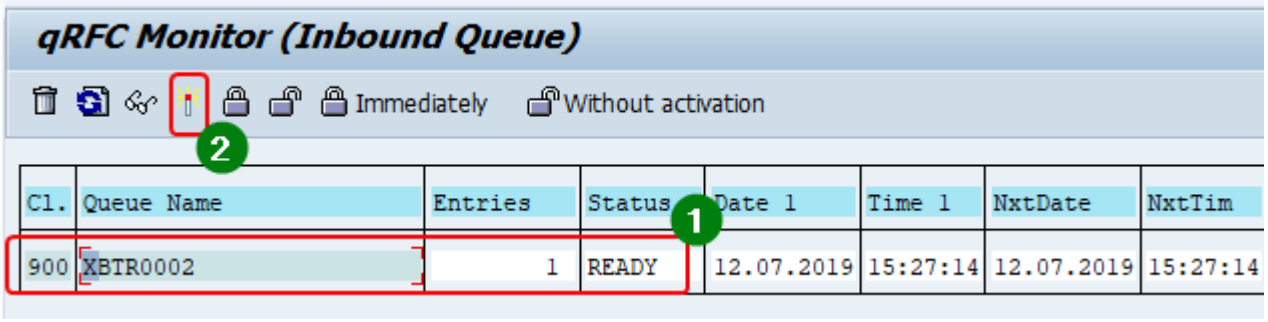
Check if inbound queues are activated using transaction SMQR. If the inbound qRFC queues are not registered, you will be able to find the queues in a "Scheduled" status in SMQ2.




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.




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 eve tech-related. I started my carrier as an ABAP developer and soon found my love for system integration wher learned SAP XI 3.0 in 2008. Playing a variety of roles from an offshore technical consultant (ABAP, PI/PO, BW,

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 ›

## 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!!

*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 *

Website

POST COMMENT



REGRESSION TEST AUTOMATION

## TAG CLOUD

ABAP | ALE/iDoc | ASMA | B2B | Base64 | Change Pointers | DDIC | EDI | Error | Example | File Adapter | FTP | Function Module | Guide | How To | HTTP | iDoc | iDoc_AAE | iFlow | Java Mapping | Message Mapping | NWA | NWDS | Overview | PI/PO | Proxy | RBDMIDOC | RFC | SAP PO 7.5 | SE11 | SE37 | SE38 | Service Interface | SFTP | SLD | sm59 | SOAP | sproxy | SQ01 | Step by Step | Troubleshoot | Variable Substitution | we02 | we20 | XML

## 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

SAP Integration Hub

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