

ABAP SPROXY & SXMB_MONI

Table of contents

1.	SPROXY and PI / PO.....	3
2.	SPROXY Screen Information	4
3.	Creating / Activating Proxy.....	6
4.	Generate Service.....	8
5.	Add Services To Favorites	8
6.	Testing The Service	9
7.	Service Usage Code Example.....	10
8.	SXMB_MONI.....	11

1. SPROXY and PI / PO

SPROXY is basically a transaction code.

SPROXY is used when there is PI/PO module in the system.

SAP PI and SAP PO are integration systems to connect SAP and non-SAP environments with each other, universal adapters that allow us to monitor, at any time, the data that comes from your systems or third-party systems.

For example, if the integrator side has created a SOAP service and the client side wants to respond to it as a rest, both the integrator and client provide the connection, the bridge between them.

At this point, SPROXY is used to create or activate an ABAP proxy.

ABAP proxies convert abap data to XML or vice versa, depending on whether they are incoming/server or outgoing/client.

Interface objects can be viewed from the integration repository with the navigation tree in SPROXY.

2. SPROXY Screen Information

- 1- When we go to the SPROXY transaction code, we are greeted by a screen like the one below. Here we can see and examine standard objects, services, namespaces, etc. from the navigation tree on the left.

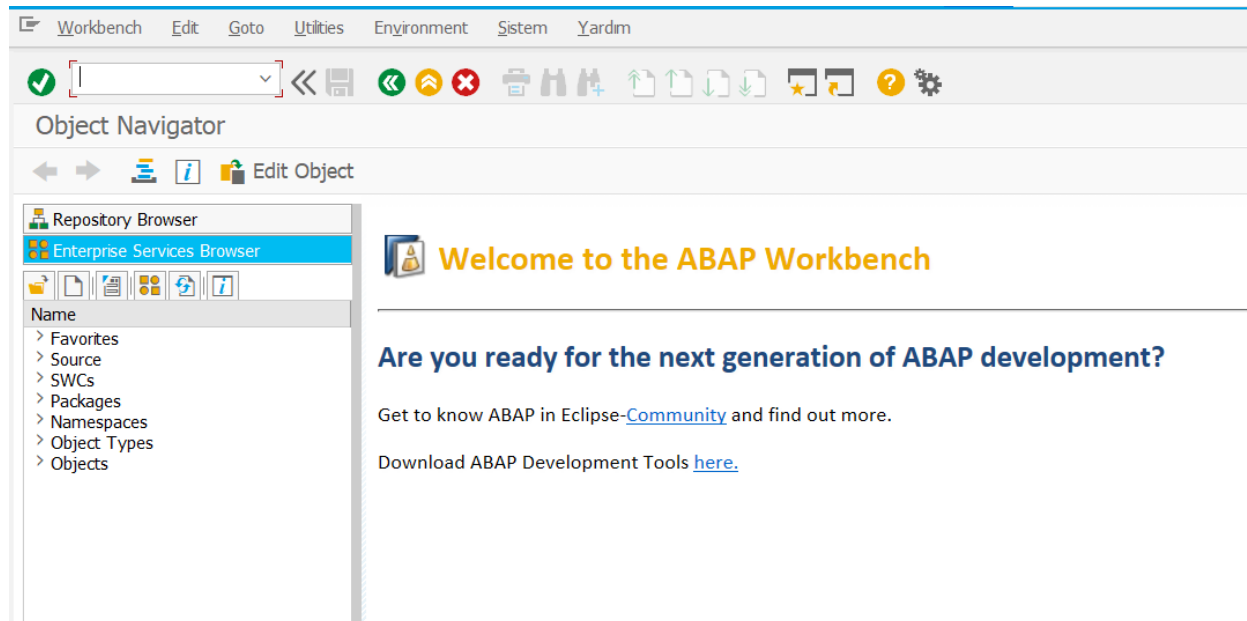


Figure 1: SPROXY Screen

- 2- While on this screen, we can see the services in the system or the service titles that are grouped under certain titles via "Source" -> "ESR" -> "SWCs". When we click on any service, the "Objects" and "Object Types" sections will appear.
- 3- In the Object Types section, we can examine the data types, service provider, service consumer and message types. In the Object section, we can see the general inbound and outbound structure of the service together.

- 4- You can find the relevant image below;

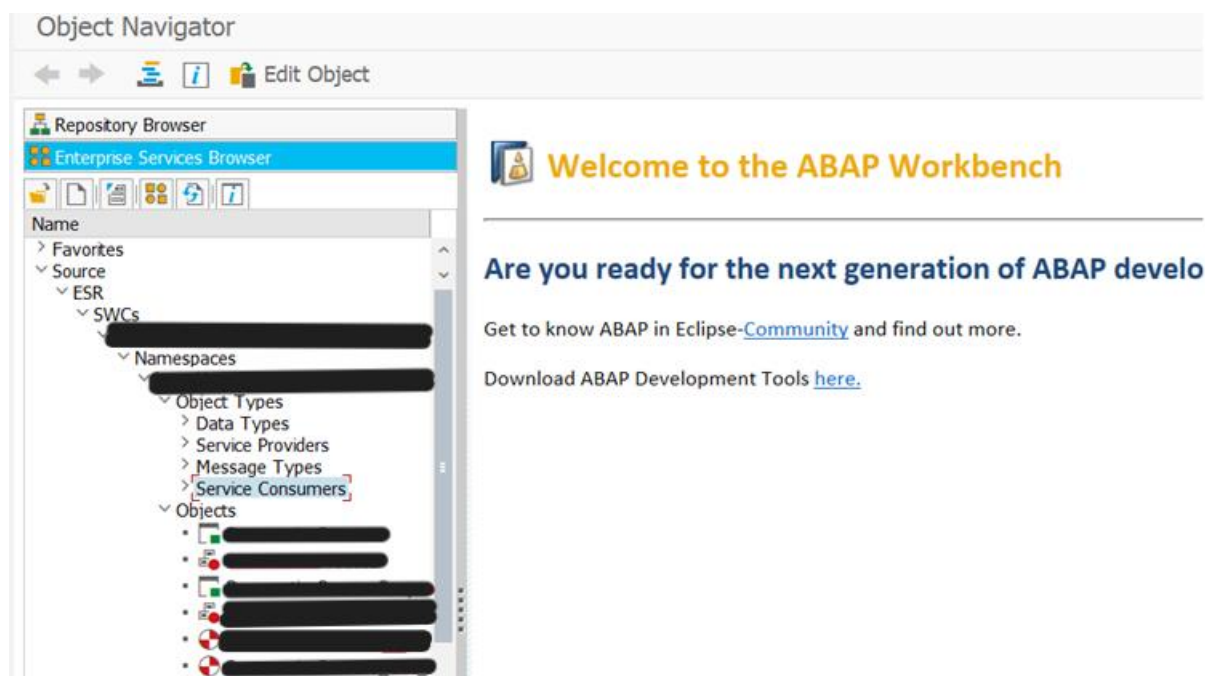


Figure 2: Subheadings Of The Service

- 5- In this section, we see the colors of the symbols for a service that has not yet been created as “red”, while the symbols of activated/used services are displayed as “green”. The phrases “Out_S” and “In_S” at the end of the service names show the structures that will be created depending on whether the service is outbound or inbound with smart code logic.

3. Creating / Activating Proxy

When we double click on the service, we are greeted by a screen like the one below. Here, the package name, request information and prefix field, if desired, are filled in and the Proxy is created when the continue button is clicked. Then, the service is activated when the save and activate button is clicked.

Enterprise Services: Create Service Provider

Transport
Finish

Enter the package for the new object or check Local Object to use \$TMP.
If already known you may enter the request to be used.
For proxy objects you have the option to specify a prefix.

☐ Local Object

Package

Talep/görev

Prefix

Back Continue Cancel Finish

Figure 3: Creating / Activating the Service

When we enter a previously created service, we are greeted by a screen like the one below;

Service Consumer [Redacted] Active

Properties External View Internal View Objects Configuration WSDL Warnings Classifications

Service Consumer

Name	[Redacted]
Namespace	[Redacted]
ABAP Object	CLAS Class
ABAP Name	[Redacted]
Prefix	[Redacted]
Source	Enterprise Services Repository
Description	Proxy Class (generated)

General Data

Package	[Redacted]
Original Language	EN İngilizce
Release Status	Not Released
Created by	[Redacted] on [Redacted]
Changed by	[Redacted] on [Redacted]

Figure 4: Service Content

In the image above, in the “Properties” tab, we can see information such as abap name, prefix, description, namespace at the top, while at the bottom we can see package, creator, date and time information.

In other tabs; we can see the objects in the “objects” tab, information about the configuration settings in the “configuration” tab, and WSDL link information in the “WSDL” tab.

4. Generate Service

When a new field is added to the service, a field is deleted, or the structure of a field is changed, the PI/PO side asks us to generate the service after performing the necessary operations. For this operation, when we right-click on the relevant service and say -> generate, save and activate it, the service is regenerated. In some cases, we can also go to the solution of deleting the service completely and recreating it. In this case, if there is a code behind the service, it is important to back it up. To answer the question of whether a code is written here, it is necessary to double-click on the “ABAP Name” in the properties tab and double-click on the Method name to check the opened section.

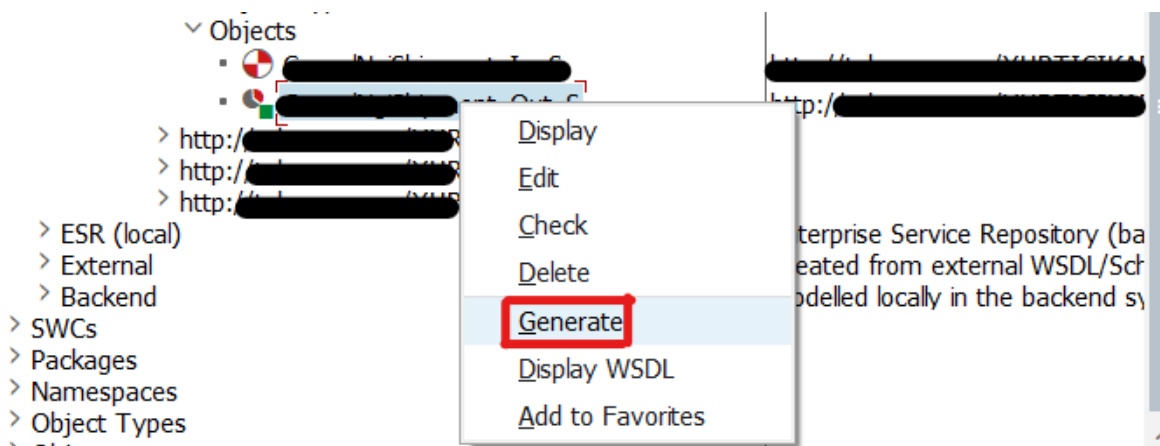


Figure 5: Generate Service

5. Add Services To Favorites

Sometimes, in projects where we work with many integrations, we can lose time while trying to find the services we are interested in. In order to prevent these situations, we can easily find the service we are looking for among our favorites by adding the services we are interested in to our favorites. For this, it will be enough to right click on the relevant service or service group -> “add to favorites”. Then, as seen in the image below, we will be able to view our favorite objects in the “Favorites” tab at the top of the navigation tree.

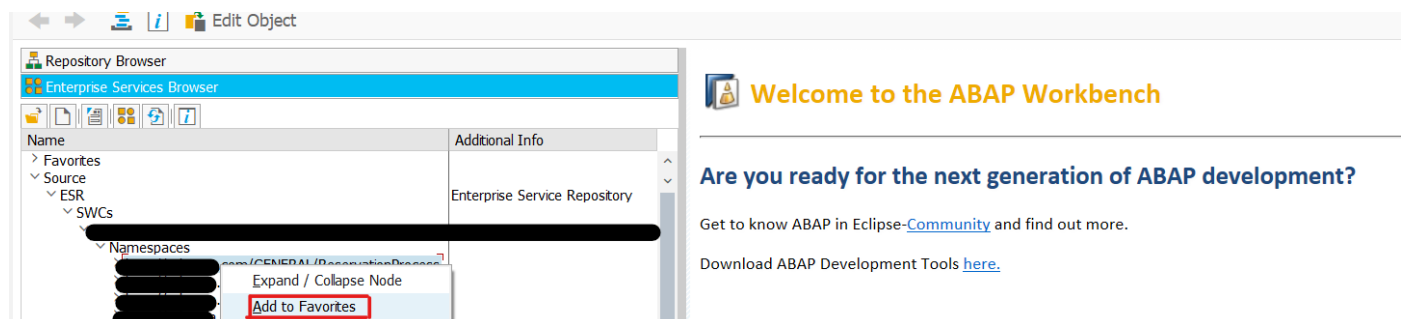


Figure 6: Add To Favorites

6. Testing The Service

When we double-click on the service, execute it with “F8” while in the “properties” tab, and press “F8” again on the screen that appears, we are greeted by a screen like the one below;



Figure 7: Sample Request View

Here we see the request structure to which the service will be sent. When we click on the marked button to make changes, the fields below will appear open to change.

After filling in the required form, when executed again with “f8”, the service will run and a response will be returned to us in a previously determined structure. If we fill it in and click the save button at the top, we will have saved the request as a variant. You can see an example of the returned response below.

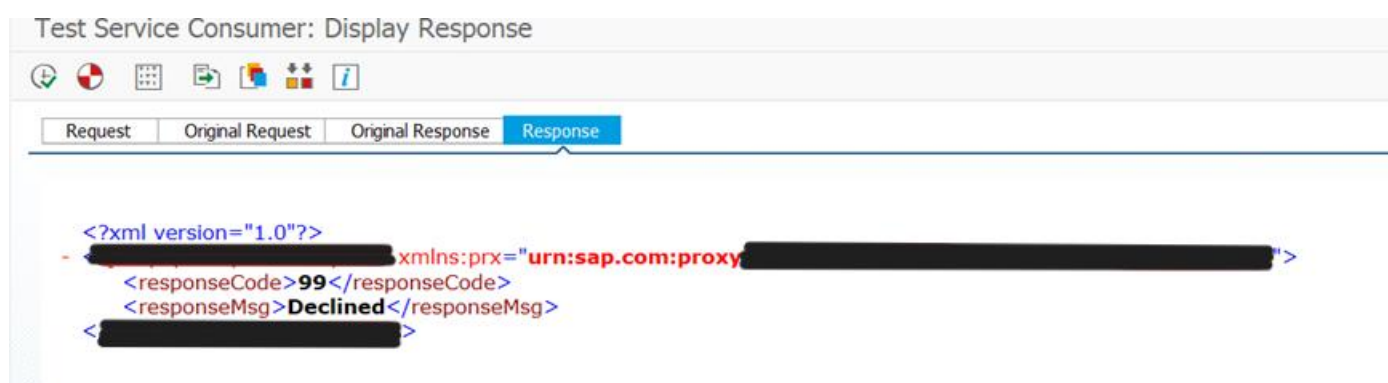


Figure 8: Sample Response View

After filling in the required form, when executed again with “f8”, the service will run and a response will be returned to us in a previously determined structure. If we fill it in and click the save button at the top, we will have saved the request as a variant. You can see an example of the returned response below.

7. Service Usage Code Example

```

*----- Request Object is Defined
DATA: ls_output TYPE <service_structure_type>.

*-----
*
* Filling the request
*
*-----

"sending the service
TRY .
  DATA(lo_proxy) = NEW <service_object>.
  lo_proxy->
><service_method>( EXPORTING output = ls_output IMPORTING input = DATA(ls_input) ).
  COMMIT WORK AND WAIT.

  CATCH cx_ai_system_fault INTO DATA(ls_ai_system_fault).
    MESSAGE ls_ai_system_fault->error_text TYPE 'E'.
ENDTRY.
  
```

8. SXMB_MONI

The SXMB_MONI event is used in SAP to monitor the processing for XML messages, track message status, and errors. XML is a formatted message that we send from one system to another.

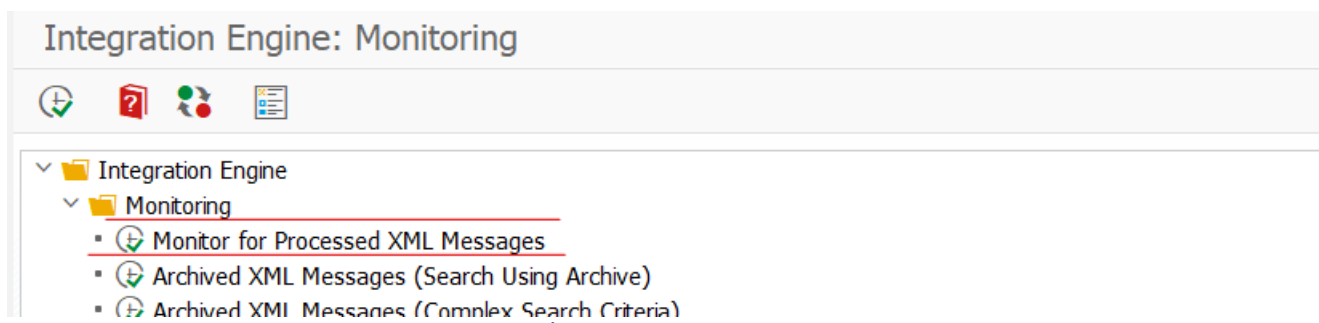


Figure 9: SXMB_MONI Screen

In the image below, you can see the selection screen. Here:

1 = Status Filter (e.g., processed, errors, scheduled)

2 = Date Filter

3 = If the 'namespace' in the proxy object includes something like ww...com/XXXX/get_proxy, you can filter it by entering 'XXXX' in this field.

4 = You can filter by entering the 'name' field in the proxy object.

5 = You can filter by entering the 'namespace' field in the proxy object.

Additionally, you can filter outbound services using the 'Sender' field and inbound services using the 'Receiver' field, as shown in the image.

Monitor for Processed XML Messages

Length of Output List: 200

Select Messages By

☒ Status Group ☐ Status

Standard Selection Criteria | Advanced Selection Criteria | User-Defined Selection Criteria

End of Execution

From Date/Time	11.12.2024	/	01:48:09
To Date/Time	12.12.2024	/	00:00:00

Sender

Party	
Component	
Interface Name	
Interface Namespace	
Operation	

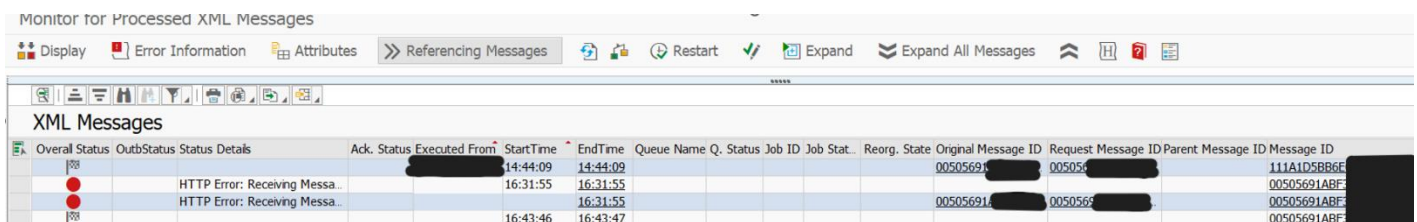
Receiver

Party	
Component	
Interface Name	
Interface Namespace	
Operation	

Figure 10: Monitor for Processed XML Messages

After entering the necessary filters and executing with F8, we see a screen like the one shown below.

Here, we can view detailed information about the relevant requests, such as their status, dates, message IDs, and many other fields.



Monitor for Processed XML Messages

Display Error Information Attributes Referencing Messages Restart Expand Expand All Messages

XML Messages

Overall Status	OutStatus	Status Details	Ack.	Status	Executed From	StartTime	EndTime	Queue Name	Q. Status	Job ID	Job Stat.	Reorg. State	Original Message ID	Request Message ID	Parent Message ID	Message ID
		HTTP Error: Receiving Messa...				14:44:09	14:44:09						00505691	00505691		111A1D58B6E
		HTTP Error: Receiving Messa...				16:31:55	16:31:55						00505691	00505691		00505691ABF
						16:43:46	16:43:47						00505691	00505691		00505691ABF

Figure 11: List of Data Communications

When we double-click on any row, we see a screen like the one shown below.

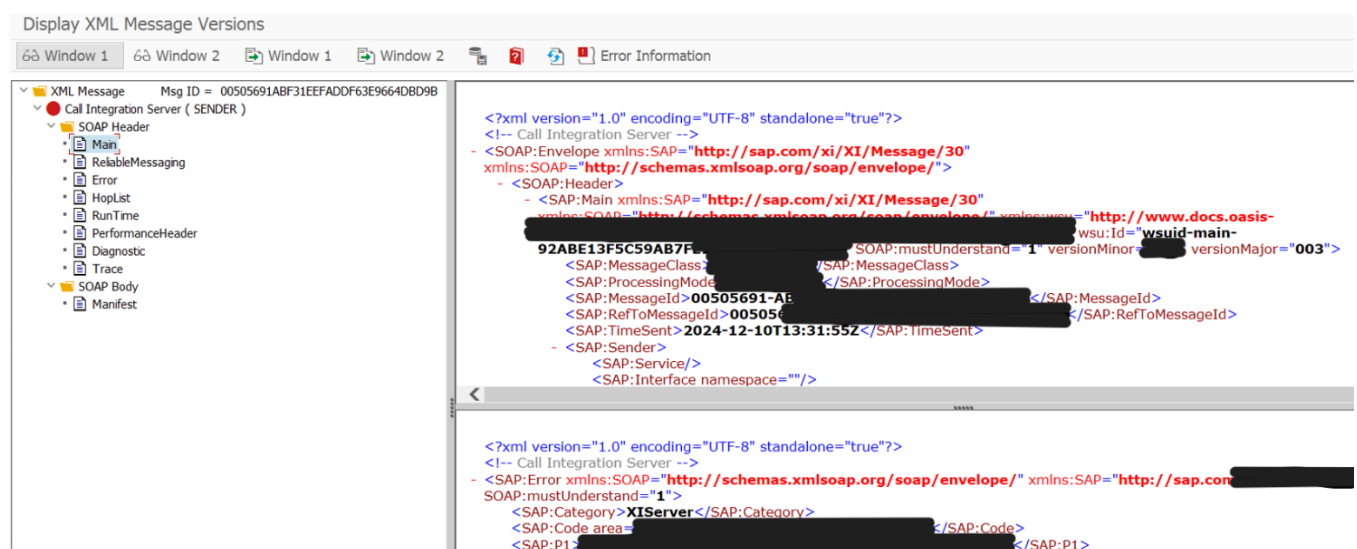
On this screen, we can view the request and response information in XML format on the right side.

On the left side:

Runtime: Displays runtime information.

PerformanceHeader: Shows how the process has progressed step by step in microseconds.

Trace: Allows us to see the step-by-step flow of the processes.



Display XML Message Versions

Window 1 Window 2 Window 1 Window 2 Error Information

XML Message Msg ID = 00505691ABF31EEFADF63E9664DBD9B

Call Integration Server (SENDER)

SOAP Header

- Main
- ReliableMessaging
- Error
- HopList
- RunTime
- PerformanceHeader
- Diagnostic
- Trace

SOAP Body

- Manifest

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<!-- Call Integration Server -->
<SOAP:Envelope xmlns:SAP="http://sap.com/xi/XI/Message/30"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <SAP:Main xmlns:SAP="http://sap.com/xi/XI/Message/30"
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wsu="http://www.w3.org/2000/09/xmldsig#">
      <SAP:MessageClass>
        <SAP:ProcessingMode>
        </SAP:ProcessingMode>
      </SAP:MessageClass>
      <SAP:MessageId>00505691-ABF31EEFADF63E9664DBD9B</SAP:MessageId>
      <SAP:RefToMessageId>00505691-ABF31EEFADF63E9664DBD9B</SAP:RefToMessageId>
      <SAP:TimeSent>2024-12-10T13:31:55Z</SAP:TimeSent>
    </SAP:Main>
  </SOAP:Header>
  <SAP:Service/>
  <SAP:Interface namespace="" />
</SOAP:Envelope>

<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<!-- Call Integration Server -->
<SAP:Error xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SAP="http://sap.com/xi/XI/Message/30">
  <SOAP:mustUnderstand="1">
    <SAP:Category>XIServer</SAP:Category>
    <SAP:Code area=" " />
    <SAP:P1>
    </SAP:P1>
  </SOAP:Error>
</SAP:Envelope>
```

Figure 12: XML View

