Example of Pega 7 SOAP Web Service Integration

Posted on **July 9, 2016** by **Bruno**

This example describes how a SOAP web service can be called from a Pega 7 application *(version 7.1.6 was used)*. The Pega 7 Integration Wizard will be used to process the WSDL file *(Web Services Description Language)* to generate the necessary Pega rules for calling the SOAP service.

Related Posts

- Creating a SOAP Web Service with Eclipse IDE and Tomcat
- Example of Pega 7 REST Service Integration with Connect REST Rule
- Accessing the Tomcat Manager Application for Pega 7 PVS Installations

Summary

1 Description of SOAP Web Service Used in this Example

For this example, a SOAP web service has been created using Java and the Eclipse Java EE IDE for Web Developers. The SOAP web service is deployed on the Apache Tomcat v7.0 web server of the Pega 7 instance *(installed from CSA/CSSA exercise system from PDN)*.

The SOAP service in this example represents a simple product catalog and provides 3 methods for searching and inserting products. Sample request and response XML for all ***three operations*** are given as follows:

`getAllProducts` *request:*

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
        xmlns:ser="https://services.pegaxchange.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:getAllProducts/>
```

```
    </soapenv:Body>
</soapenv:Envelope>
```

`getAllProducts` *response:*

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
        xmlns:xsd="https://www.w3.org/2001/XMLSchema"
        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <getAllProductsResponse xmlns="https://services.pegaxchange.com">
         <getAllProductsReturn>
            <category>Electronics</category>
            <id>1</id>
            <name>Keyboard</name>
            <unitPrice>29.99</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Electronics</category>
            <id>2</id>
            <name>Mouse</name>
            <unitPrice>9.95</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Electronics</category>
            <id>3</id>
            <name>17" Monitor</name>
            <unitPrice>159.49</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Hardware</category>
            <id>4</id>
            <name>Hammer</name>
            <unitPrice>9.95</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Hardware</category>
            <id>5</id>
            <name>Slot Screwdriver</name>
            <unitPrice>7.95</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Books</category>
            <id>6</id>
            <name>The British Invasion of Java</name>
            <unitPrice>11.39</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Books</category>
            <id>7</id>
            <name>A House in Bali</name>
            <unitPrice>15.99</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Books</category>
            <id>8</id>
            <name>An Alaskan Odyssey</name>
            <unitPrice>799.99</unitPrice>
         </getAllProductsReturn>
         <getAllProductsReturn>
            <category>Electronics</category>
            <id>9</id>
            <name>LCD Projector</name>
            <unitPrice>1199.19</unitPrice>
         </getAllProductsReturn>
      </getAllProductsResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

`insertProduct` *request:*

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
          xmlns:ser="https://services.pegaxchange.com">
   <soapenv:Header/>
```

2017
- Using a Page List for a Drop the Options with a Data Tra May 11, 2017
- How to Run Pega 7 Exercis with Oracle VM VirtualBox A
- Installing a Desktop GUI on System Virtual Machine wit 2017
- How to Connect to External Server 2016 Database from 19, 2017
- How to Connect to External Edition DB from Pega 7.1.9
- Example of Pega 7 REST Se with Connect REST Rule Oc
- Setup of Java EE Developm with Eclipse Neon and Tom 2016
- Creating a REST Web Servi Neon, Tomcat 9, JAX-RS Je Jackson August 11, 2016

CATEGORIE
- Administration (9)
  - System Debugging (2)
  - System Tuning and Mai
- Application Design (1)
  - Circumstancing (1)
- Automating Business Polic
  - Activities (2)
  - Automating Decisions (
  - Data Transforms (2)
- Data Model (3)
  - Controlling Data with Pr
  - Source Your Data with D
- Data Tables (3)
- File Service (1)
- Integration (9)
  - Configuring a REST Co

```
   <soapenv:Body>
      <ser:insertProduct>
         <ser:product>
            <ser:category>Books</ser:category>
            <ser:id>10</ser:id>
            <ser:name>A History of Western Music</ser:name>
            <ser:unitPrice>17.95</ser:unitPrice>
         </ser:product>
      </ser:insertProduct>
   </soapenv:Body>
</soapenv:Envelope>
```

`insertProduct` *response (SUCCESS):*

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsd="https://www.w3.org/2001/XMLSchema"
          xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <insertProductResponse xmlns="https://services.pegaxchange.com"/>
   </soapenv:Body>
</soapenv:Envelope>
```

`searchById` *request:*

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
         xmlns:ser="https://services.pegaxchange.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:searchById>
         <ser:id>10</ser:id>
      </ser:searchById>
   </soapenv:Body>
</soapenv:Envelope>
```

`searchById` *response (SUCCESS):*

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
         xmlns:xsd="https://www.w3.org/2001/XMLSchema"
         xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <searchByIdResponse xmlns="https://services.pegaxchange.com">
         <searchByIdReturn>
            <category>Books</category>
            <id>10</id>
            <name>A History of Western Music</name>
            <unitPrice>17.95</unitPrice>
         </searchByIdReturn>
      </searchByIdResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

`searchById` *response (FAIL):*

```
<soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
          xmlns:xsd="https://www.w3.org/2001/XMLSchema"
          xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <soapenv:Fault>
         <faultcode>soapenv:Server.userException</faultcode>
         <faultstring>java.lang.Exception: No product found with id 11</faultstring>
         <detail>
            <ns1:hostname xmlns:ns1="https://xml.apache.org/axis/">prpc</ns1:hostname>
         </detail>
      </soapenv:Fault>
   </soapenv:Body>
</soapenv:Envelope>
```

- Before continuing to step 2, it is a good idea to confirm the availability of the SOAP service endpoint URL and WSDL file. In this example, the endpoint URL of the SOAP service is:

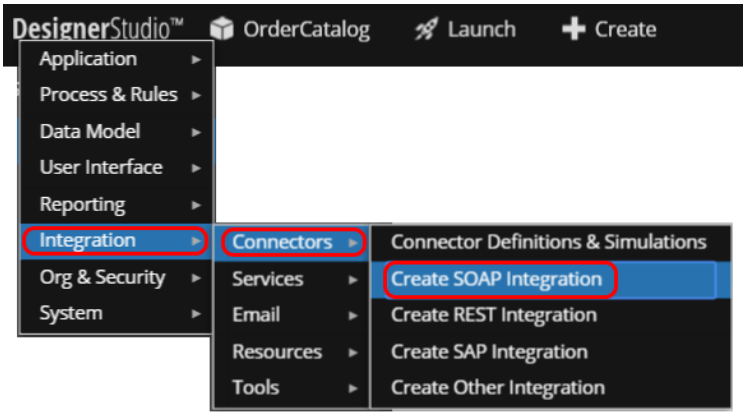
```
https://prpc:8080/ProductCatalogSOAPService/services/ProductCatalogServiceImpl
```

- …and the WSDL file is accessed by adding `?wsdl` to the endpoint URL:

```
https://prpc:8080/ProductCatalogSOAPService/services/ProductCatalogServiceImpl?wsdl
```

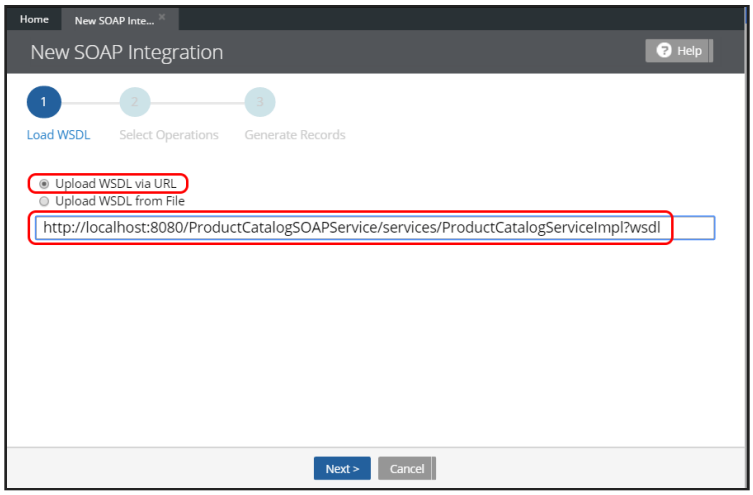
2 Creating a new SOAP Connector Integration in Pega 7

- In the Designer Studio navigate to ***Designer Studio*** > ***Integration*** > ***Connectors*** > ***Create SOAP Integration***.

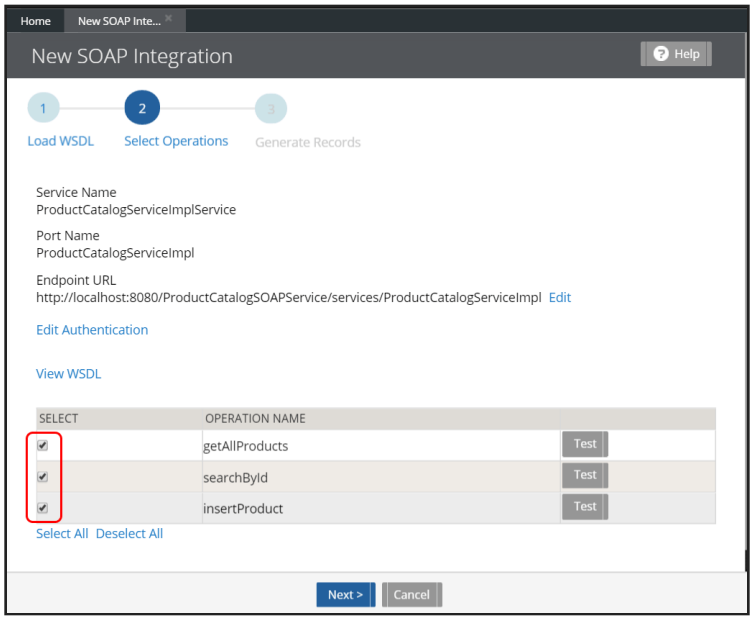


- In the first step of the ***New SOAP Integration*** form, the WSDL file needs to be specified.
- This can either be done by using the URL of the WSDL or by uploading the WSDL file manually.
- Here, the URL is used and the SOAP web service has been deployed to the Pega 7 Tomcat web server.
- See this post on [how to access the PRPC Tomcat manager application](#).
- The URL to access the WSDL from the Pega 7 PRPC web server is therefore:

```
https://localhost:8080/ProductCatalogSOAPService/services/ProductCatalogServiceImpl?wsdl
```



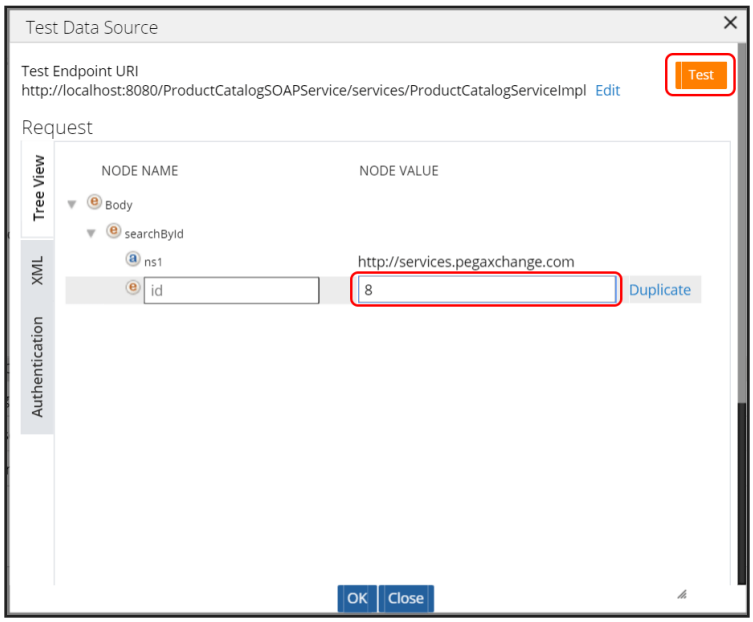
- Click on ***Next >*** to continue. The wizard will download and process the WSDL file and show all available service operations on the next step page.
- ***Select the operations*** for which PRPC should generate integrations by clicking on the check box next to the operation name. In the below example, all available operations are selected.



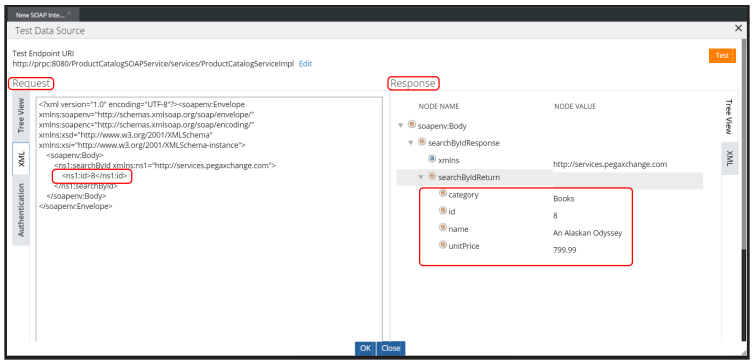
Test Data Source

- Individual operations can be tested in this step by clicking on the ***Test*** button for a given operation.
- In the screen shot below, the operation `searchById` is tested.
- On the first screen, the request parameters for the operation can be specified.

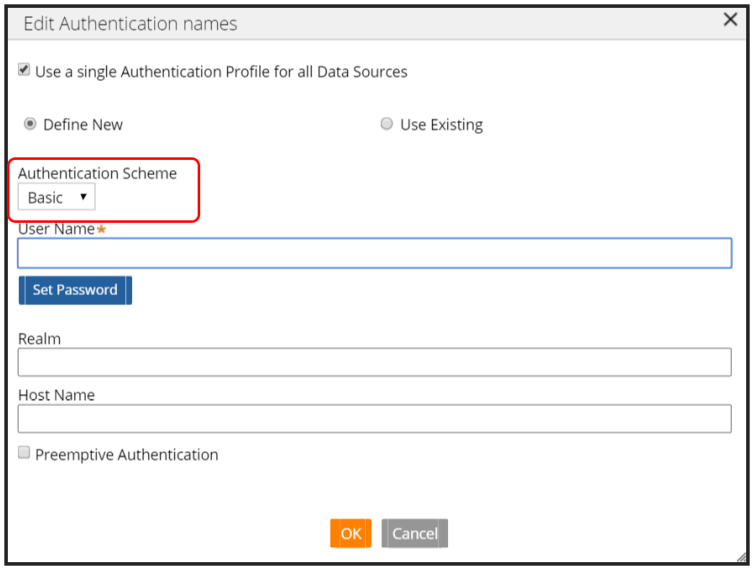
- This method has one input parameter, named `id`, to search for a product in the catalog by unique ID.



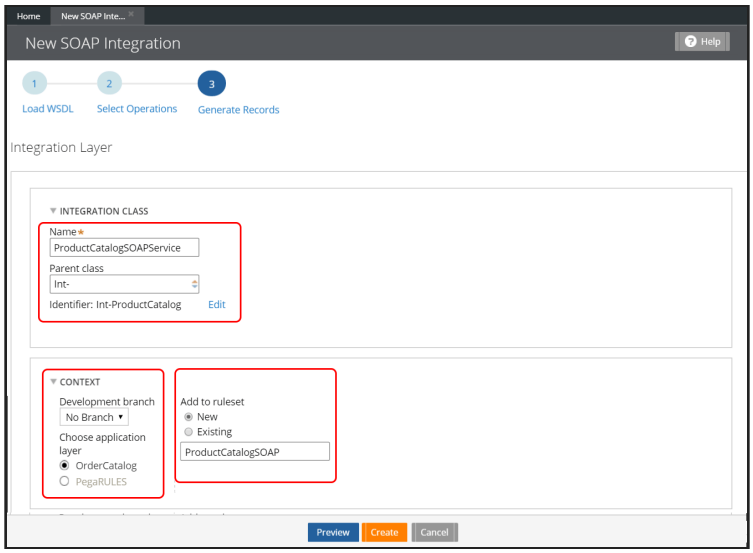
- Clicking on the ***Test*** button will result in a call to the operation `searchById` with `id=8`.
- The result of the service call is shown in the ***Response*** area in the right side of the screen.
- The request XML can also be edited directly in the left-hand side area of the screen and the ***Test*** button will trigger another new service call.



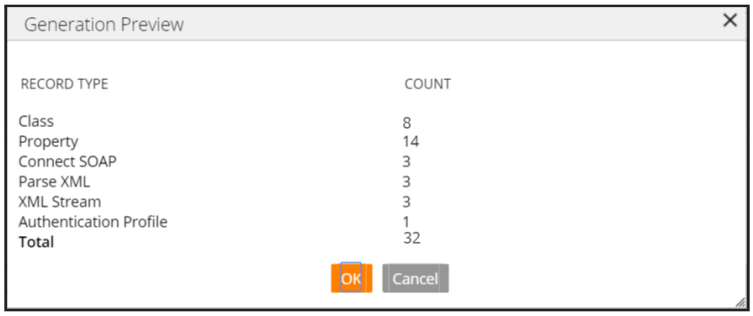
- Click on the ***Ok*** button at the bottom of the screen to close the ***Test Data Source*** dialog.
- ***Optional***: The ***Edit Authentication*** link on the main screen of step 2 allows to configure authentication, if needed.
- See the Pega 7 help page Create SOAP Integration for more information on configuring web service authentication.
- The following 3 authentication schemes are supported: **`Basic`**, **`NTLM`** and **`OAuth`**
- In this example, the **`Basic`** authentication scheme is selected.



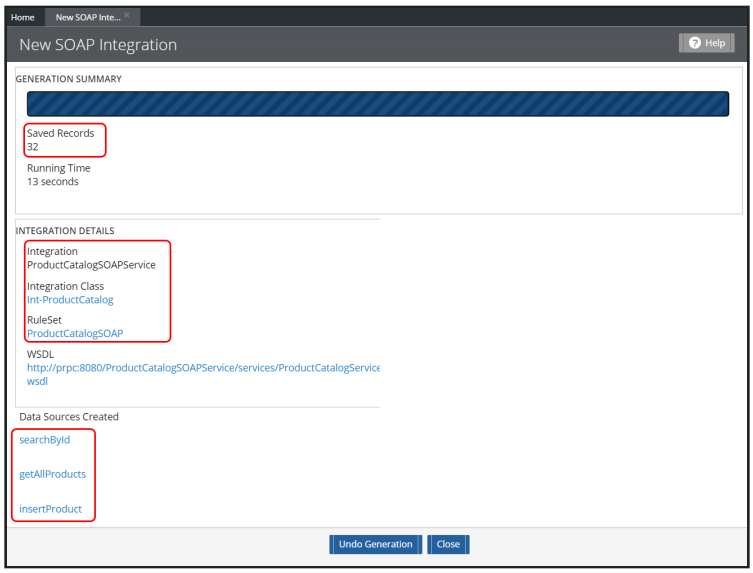
- No authentication has been configured for the SOAP service integration in this example service.
- Back on the main screen of step 2, click on ***Next*** to continue to step 3 of the wizard.
- On the ***Integration Layer*** screen, specify the name of the integration class and, if needed, set a custom name for the identifier of the integration class.
- In this case, the identifier of the integration class is `Int-ProductCatalog`.



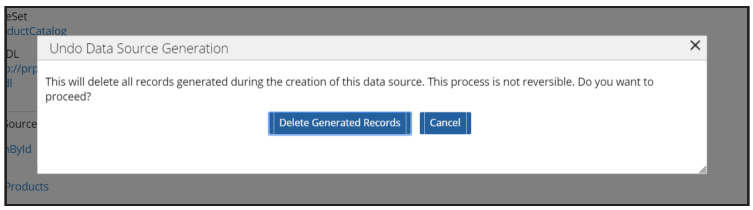
- The ***Context*** section can be used to specify where the generated rules will be stored.
- Select an existing ruleset or check the ***New*** checkbox to let the wizard generate a new ruleset for this service integration *(this is the recommended option)*.
- Before continuing, the ***Preview*** button can be used to see how many rules the wizard will generate.
- In this case, a total of ***32*** rules will be generated.



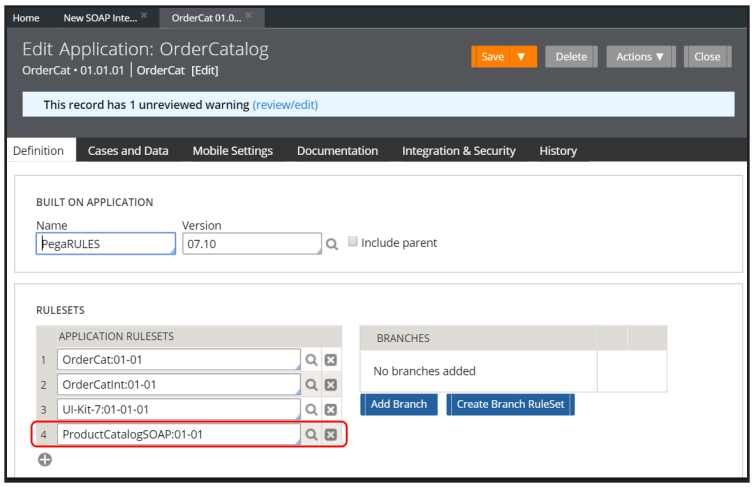
- Click on the ***Create*** button to start the automatic rule generation by the wizard.
- This process may take a few minutes. After completion, a confirmation screen will show a summary of the automatic rule generation.
- In the below example, 32 rules and 3 data sources *(one Connect SOAP rule for each web service operation)* were created in 16 seconds.



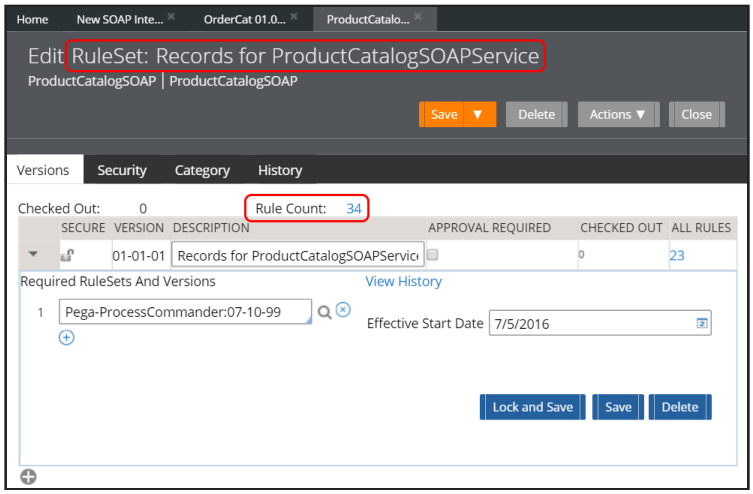
- All rules created during the generation can be removed from the system by clicking on the ***Undo Generation*** button.
- A confirmation dialog will appear and clicking on ***Delete Generated Records*** will trigger the undo process.



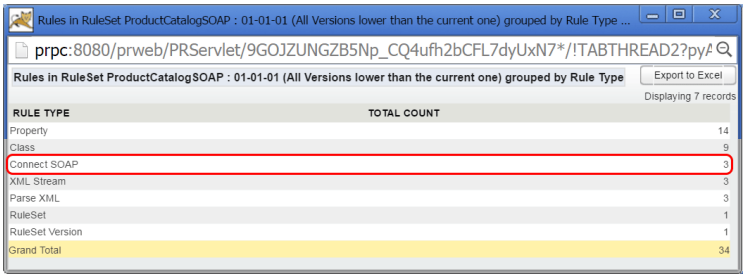
- On the ***Generation Summary*** screen, click on ***Close*** to close the integration wizard.
- In step 3 of the wizard, titled ***Integration Layer***, the application layer was selected in the context section.
- The new service ruleset should have been automatically added to the application rulesets as shown below.



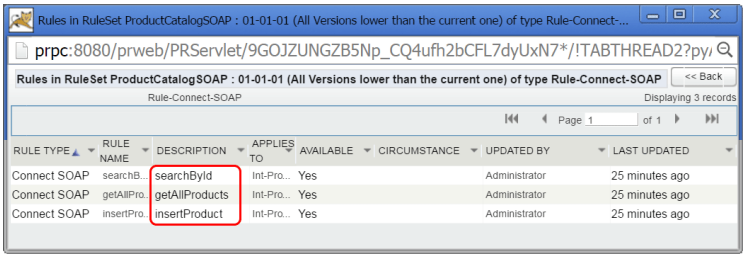
- ***Note:*** For proper referencing of the generated rules:
- Add the service ruleset, here **`ProductCatalogSOAP:01-01`** as a prerequisite to the integration ruleset, here **`OrderCatInt:01-01`**.
- Add the service ruleset as a prerequisite to the **`UI-Kit-7:01-01-01`** ruleset *(use the magnifying glass next to it's name)*.
- Use the Designer Studio search field to open the **`ProductCatalogSOAP`** ruleset.
- All of the generated rules in the ruleset can be viewed by clicking on the number next to the ***Rule Count*** label.



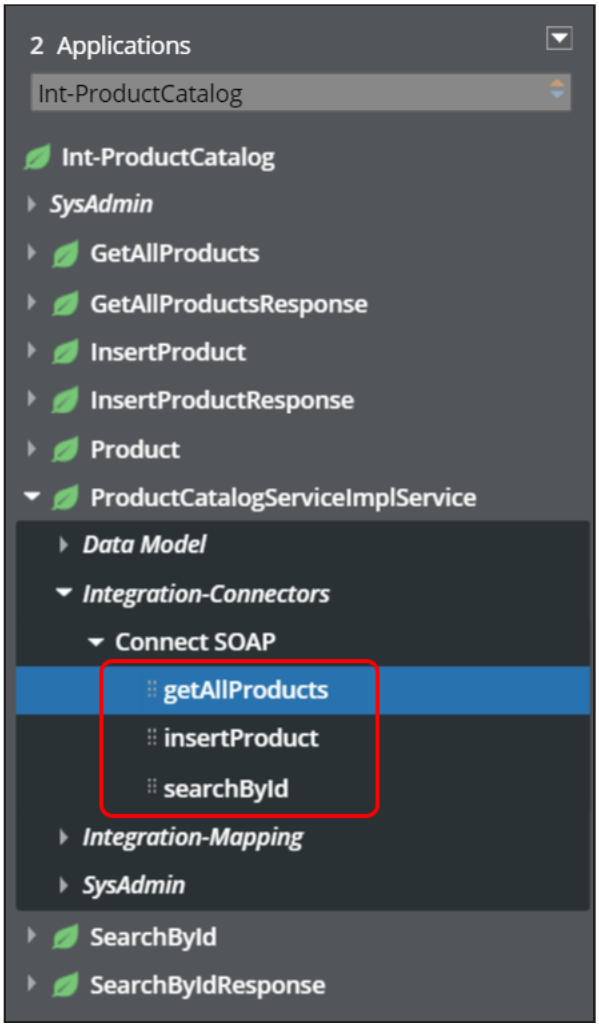
- The ***Rules in RuleSet…*** table view allows to drill down into specific rule types.



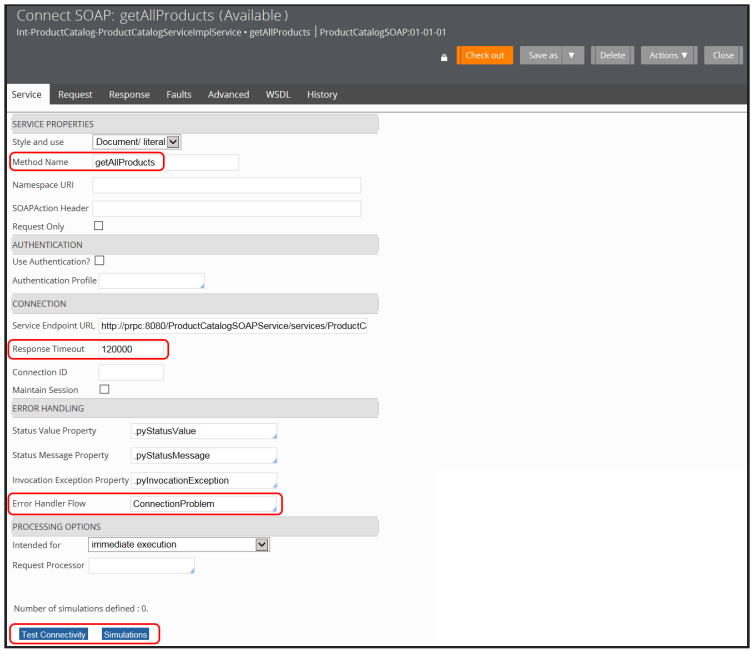
- Below, all 3 generated Connect SOAP rules are shown. There is one rule for each SOAP service operation.
- A particular Connect SOAP rule type instance can be opened by clicking on the rule name in table.



- The ***Application Explorer*** can also be used to open the individual SOAP service integration rules.

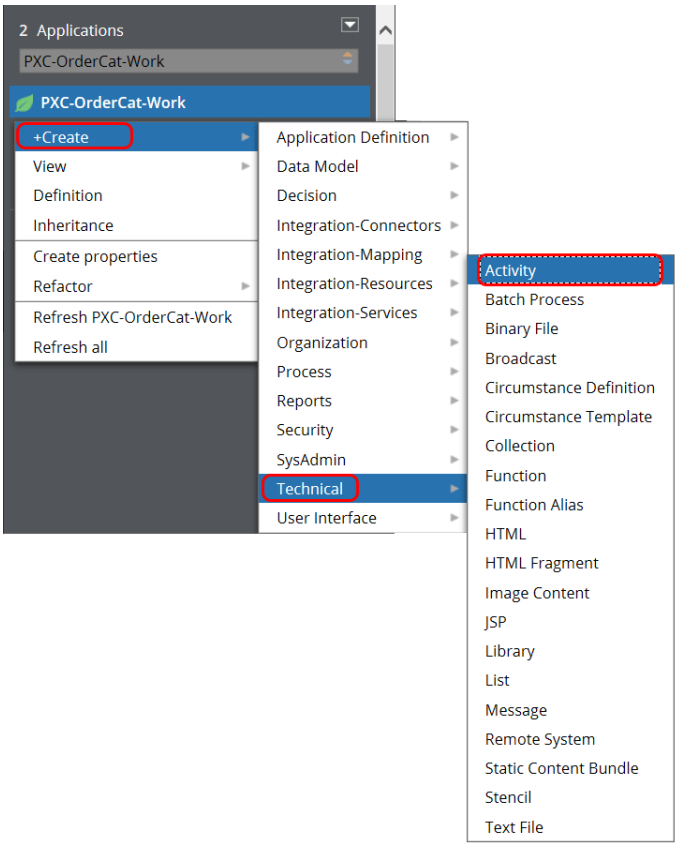


- ***Note:*** If opening the generated rules fails with a message saying that Pega could not find the rule, log-out and log back into the Designer Studio. In ***Pega 7.1.6***, the Connect SOAP, Parse XML and XML Stream rules can only be opened using Internet Explorer.
- The Connect SOAP rule for the web service operation `getAllProducts` can be opened by clicking on the name in the Application Explorer. It will open in a new window.
- The ***Service*** tab of the Connect SOAP rule allows the configuration of response timeout, error handler flow and the service endpoint URL.

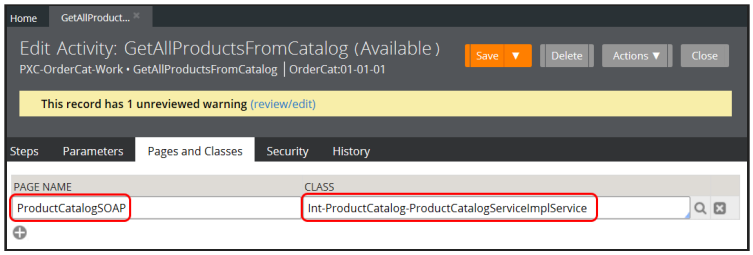


3 Calling the External SOAP Web Service from an Activity

- In the Designer Studio, create a new activity in the application `-Work` class for calling the SOAP web service.
- Here the work class name is `PXC-OrderCat-Work`.

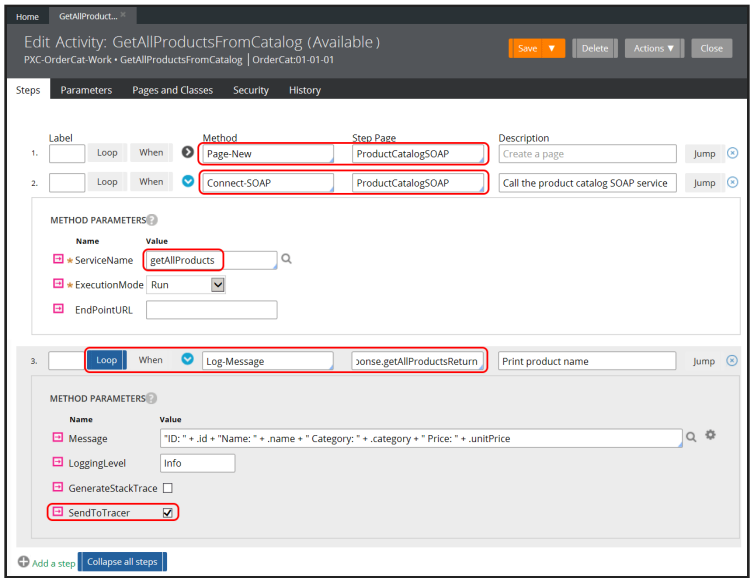


- In this case, the activity will retrieve all products from the service using the Connect SOAP rule for `getAllProducts`.
- On the ***Pages and Class*** tab, set a page name referring to the generated `...ImplService` class:

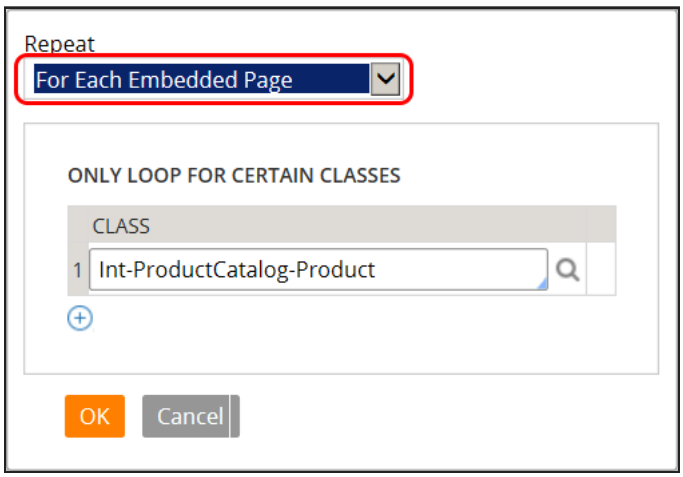


On the ***Steps*** tab, the following steps needs to be added:

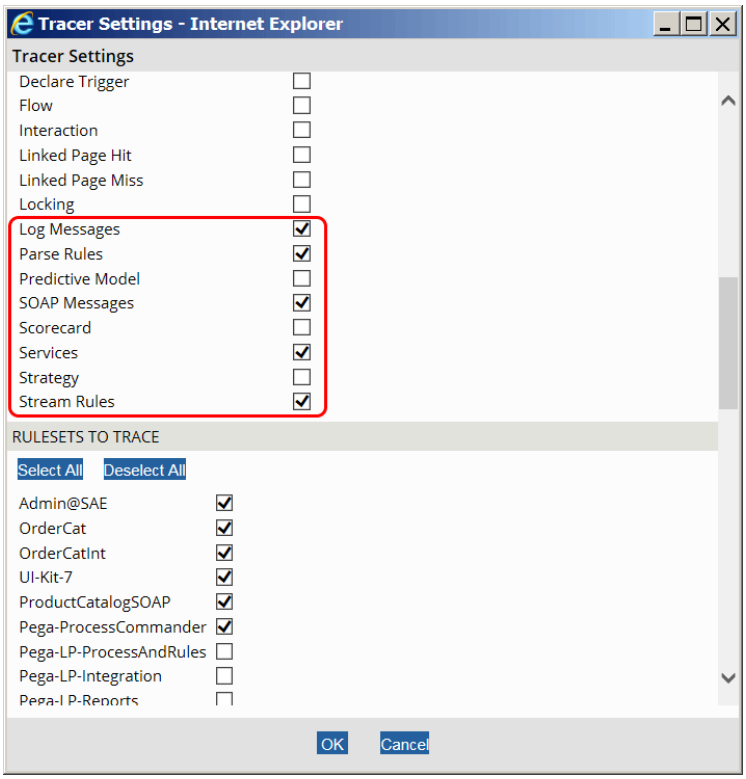
- **`Step 1`**: Create a new page instance of the service implementation using the Page-New activity method.
- **`Step 2`**: Call the SOAP web service using the Connect-SOAP method.
- **`Step 3`**: The products received from the service are logged and sent to the tracer using the Log-Message method and by enabling it's **`SentToTracer`** parameter as shown in step 3 of the activity.



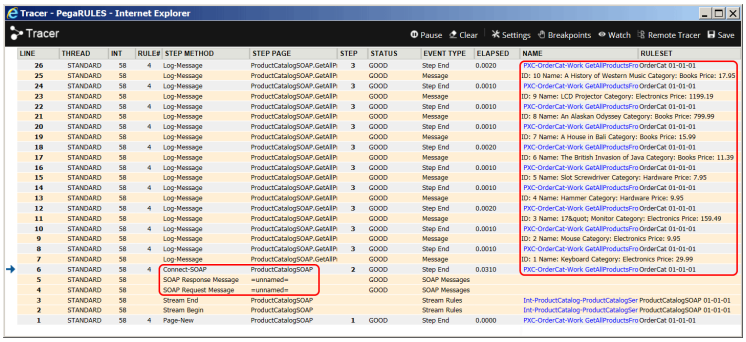
- Note that the 3rd step uses an activity Loop to iterate over the products in the response.
- **`ProductCatalogSOAP.GetAllProductsParametersResponse.getAllProductsReturn`** is the step page for the 3rd step in this example.
- Refer to the response XML for the **`getAllProducts`** operation *(see section 1)* to understand the step page structure.
- The loop iterates over a page list where each item is a page of **`Int-ProductCatalog-Product`**.
- The configuration of the loop step looks like this:



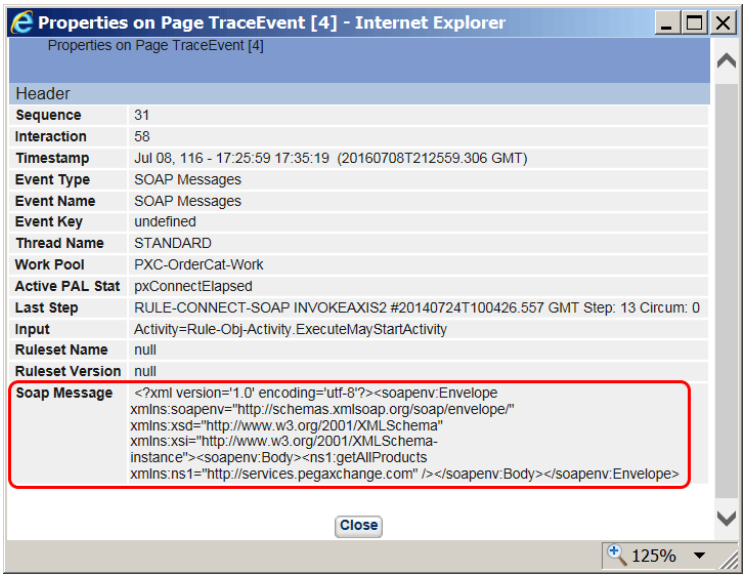
- The activity can be run by clicking on ***Actions*** > ***Run***.
- Open the ***Tracer*** to monitor the SOAP request and response XML.
- In ***Tracer Settings***, enable these items: `Log Messages`, `SOAP Messages`, `Parse Rules`, `Stream Rules` and `Services`
- Also make sure that the rulesets containing the rules for tracing are checked under ***RULESETS TO TRACE***.
- Here, the SOAP integration rules are contained in `ProductCatalogSOAP`.



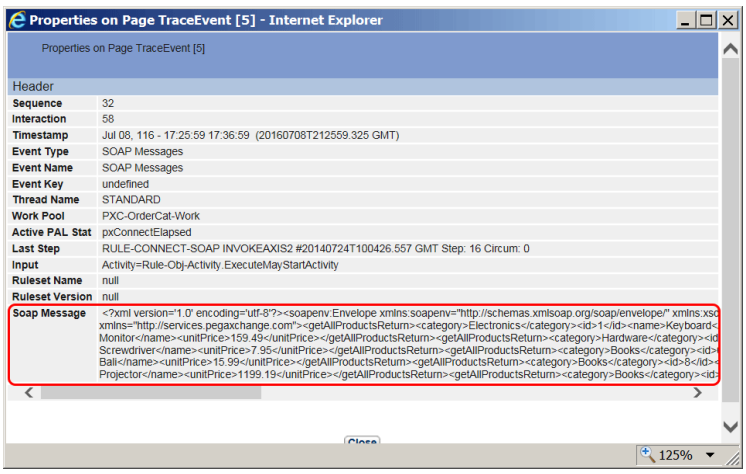
- The Tracer will show the log messages containing the product data received from the service.



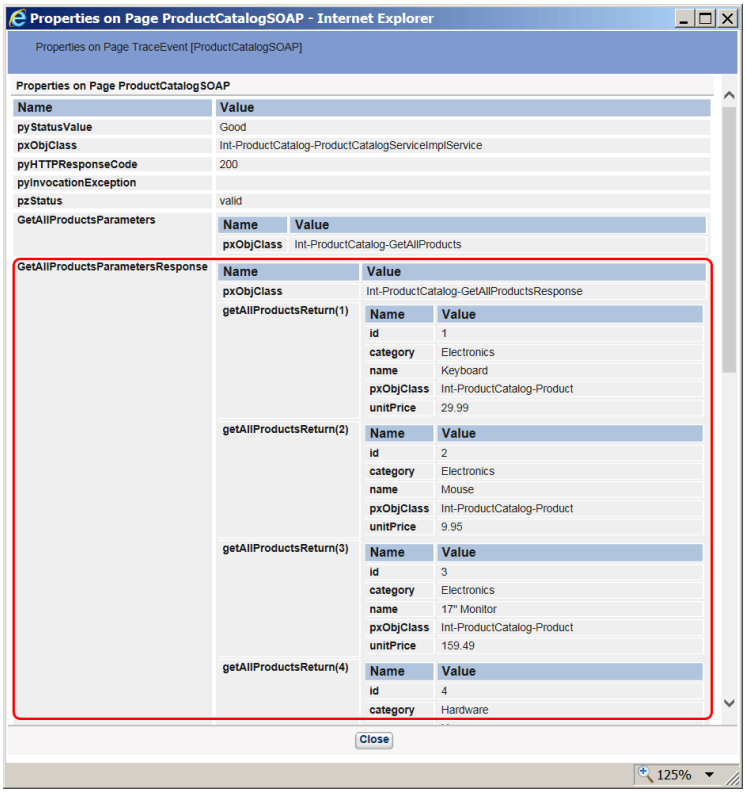
- Clicking on the `SOAP Request Message` item of the Connect-SOAP step method in the tracer provides a view of the request XML sent from the Pega 7 SOAP connector to the web service.



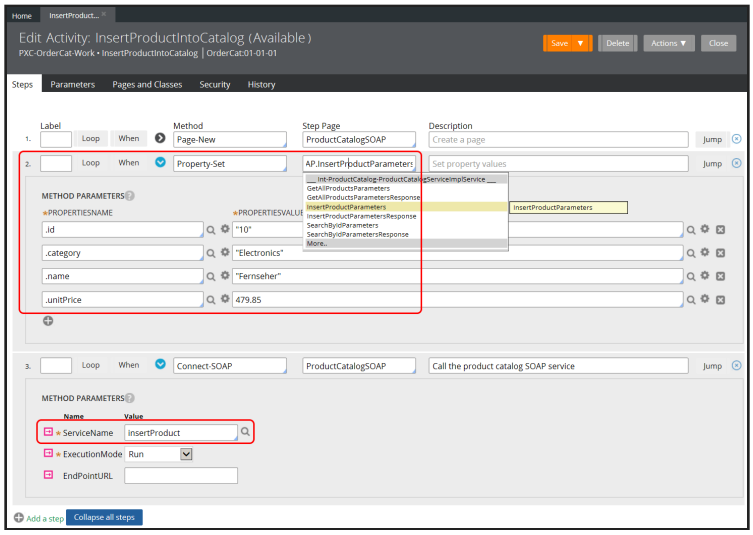
- Likewise, clicking on the **`SOAP Response Message`** item of the Connect-SOAP step method in the tracer provides the response XML received by the SOAP connector from the web service.



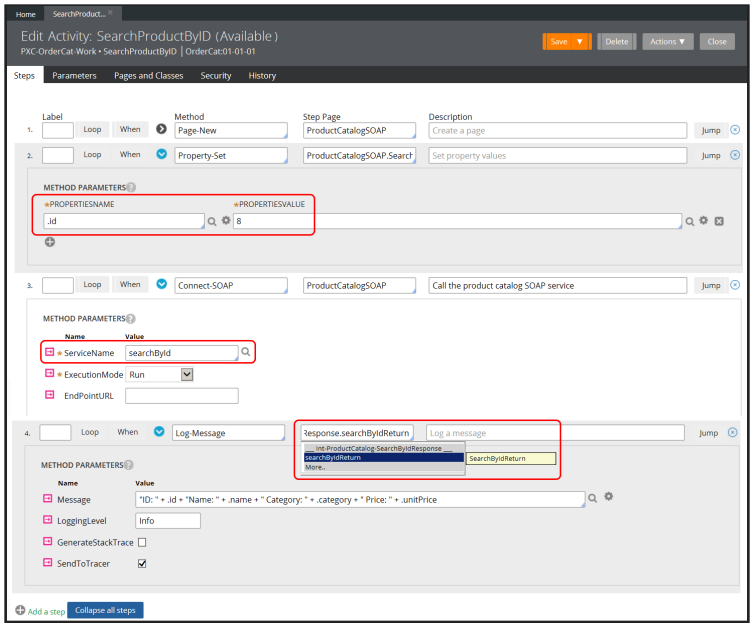
- The service connector page, specified on the Pages and Classes tab of the activity, can be viewed by clicking on the step page of the Connect-SOAP step method in the tracer.
- In this case, the step page is named **`ProductCatalogSOAP`** and it contains the products in an embedded page named **`GetAllProductsParametersResponse`**, which contains a page list named **`getAllProductsReturn`**.
- The service connector page also contains the HTTP status code in a property named **`pyHTTPResponseCode`**.
- The status code **`200`** indicates a successful service call.
- The service connector page will also contain error information if the service call fails.



- The steps of the activity for ***inserting a product*** uses the `insertProduct` Connect SOAP rule.
- Note that for this request, input parameters have to be set. See activity step 2 in the screen shot below.
- The step page for the request parameters is `ProductCatalogSOAP.InsertProductParameters.product`.
- It is a single page property of the type `Int-ProductCatalog-Product`.

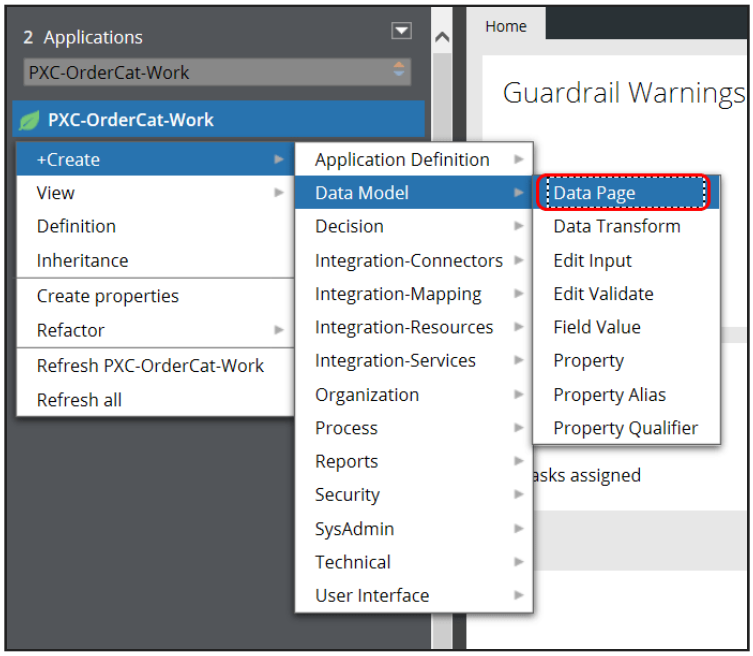


- The steps of the activity for ***searching a product*** using the `searchById` Connect SOAP rule is shown below.
- This request requires only the ***id*** of the product to search for.

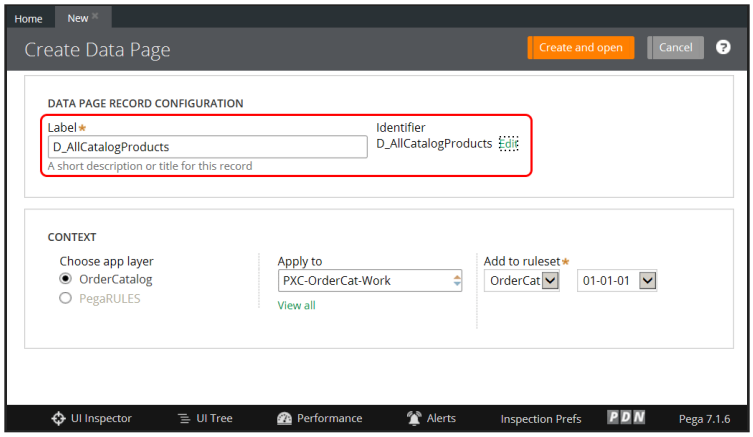


4 Calling the External SOAP Web Service from a Data Page

- In the Application Explorer, right click on the application's `-Work` class and select ***+Create*** > ***Data Model*** > ***Data Page***. Here the class name is `PXC-OrderCat-Work`.

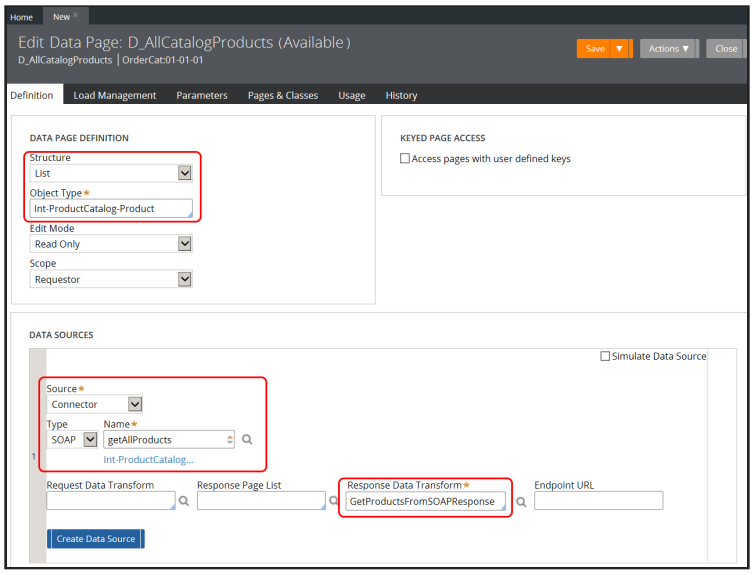


- Set an appropriate name and ***Apply to*** class and click on ***Create and open***.

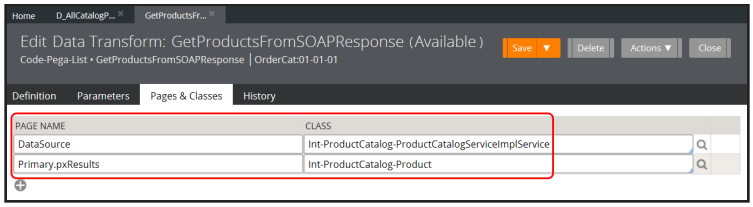


- On the ***Definition*** tab of the data page, configure it as follows:

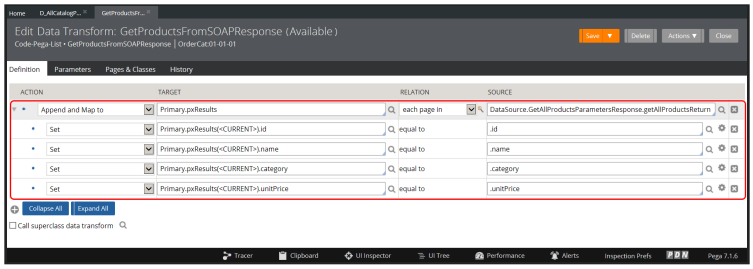
  Structure: `List`
  Object Type: `Int-ProductCatalog-Product` *(…generated by the integration wizard)*
  Source: `Connector`
  Type: `SOAP`
  Name: `getAllProducts` *(…Connect-SOAP rule generated by integration wizard)*
  Response Data Transform: `GetProductsFromSOAPResponse` *(created separately, see further below)*



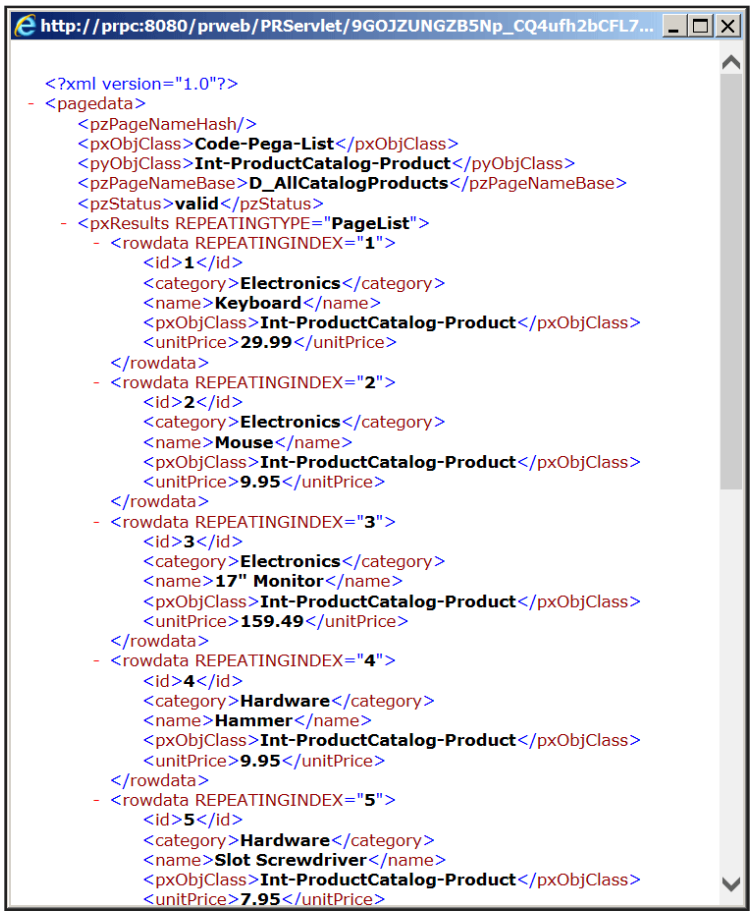
- The `GetProductsFromSOAPResponse` data transform's ***Pages and Classes*** tab looks as follows:



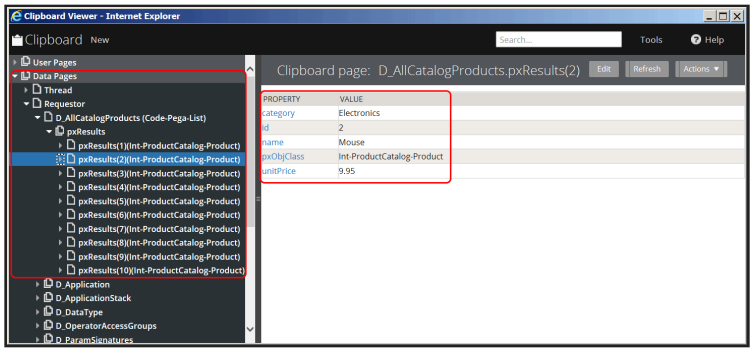
- The ***Definition*** tab, linking the service response to the data page list structure looks like this *(… click on image to enlarge)*:



- The data page can be unit tested by running it via ***Actions*** > ***Run***.
- Once execution is completed, the data page's XML will be shown:



- In addition, the data page can be examined on the clipboard with the Clipboard Viewer.
- Expand the ***Data Pages*** > ***Requestor*** nodes to view the `D_AllCatalogProducts` data page.
- In this case, the scope of the data page is ***Requestor***, so it is located under that sub-node on the Clipboard.
- The data page has a list-type property named `pxResults` of Code-Pega-List, where each item is a single page containing a product as per `Int-ProductCatalog-Product`.



This concludes the SOAP integration example in Pega 7. Please feel free to post comments or suggestions for improvement. Also, see the related posts Creating a SOAP Web Service with Eclipse IDE and Tomcat for an example of creating a SOAP service with the Eclipse Java EE IDE and Example of Pega 7 REST Service Integration with Connect REST Rule for consuming a REST service from Pega 7.

This entry was posted in **Configuring a SOAP Connector** and tagged **Pega 7 Activity Connect-SOAP**, **Pega 7 Connect-SOAP**, **Pega 7 Create SOAP Integration**, **Pega 7 New SOAP Integration**, **Pega 7 SOAP Integration Wizard**, **Pega 7 SOAP Service Integration**, **Pega 7 Tracer SOAP Message** by **Bruno**. Bookmark the **permalink**.

10 THOUGHTS ON "EXAMPLE OF PEGA 7 SOAP WEB SERVICE INTEGRATION"

pasupathi on **July 16, 2016 at 6:48 pm** said:

Clearly explained how to use Connect SOAP is really very helpful thanks for your effort and published
and one more request could you please explain Service SOAP (Integration-services).

Log in to Reply

Rose on **July 26, 2016 at 9:38 am** said:

Thank you for the SOAP integration , Could you please include a persist case and create pdf examples as well ?

Thank you in advance ..!

Log in to Reply

Bruno on **September 13, 2016 at 8:40 pm** said:

Hi, yes, create PDF sounds like an interesting topic. Will do it when I have time.

Log in to Reply

naren1993 on **September 20, 2016 at 8:27 am** said:

very good topics discussing by mr bruno we want you to discuss more topics which will help us more

Log in to Reply

Bruno
on **September 20, 2016 at 2:13 pm** said:

> Thanks for the feedback! I am currently working on an example for a REST service (JSON payload) integration. It should be done soon. What other topics do you think would be good to cover?
>
> Log in to Reply

rahulD on **February 25, 2017 at 2:11 pm** said:

Hi Bruno,

I was trying to import the created wsdl in Pega SOAP Service Integration wizard but I'm getting "SOAP Service failed. Connection refused" while testing the Operation:searchById.

WSDL: http://localhost:8080/ProductCatalogSOAPService/services/ProductCatalogServiceImpl?wsdl

The Webservice is deployed in Apache Tomcat7 server and is running.

Could you please help.

Thanks!

Log in to Reply

> Bruno
> on **February 25, 2017 at 4:43 pm** said:
>
> Hello Rahul,
>
> Did you deploy the web service on the Tomcat 7 server that is running Pega 7 (i.e. on the virtual machine)? Or is the Tomcat 7 server for the web service running on your host machine?
>
> Log in to Reply

rahulD on **February 25, 2017 at 5:52 pm** said:

Hi Bruno,

I've now deployed the .war into the Tomcat 7 Server running Pega 7 VM.

Now, I'm getting HTTP Status 500 – Servlet.init() for servlet AxisServlet threw exception, Root Cause: java.lang.UnsupportedClassVersionError:

com/pegaexchange/beans/Product : Unsupported major.minor version 52.0 (unable to load class com.pegaexchange.beans.Product)

URL: http://prpc:9080/ProductCatalogSOAPService/services/ProductCatalogServiceImpl

I think it something to do with the JDK versions of Tomcat 7 on VM and the one used in Eclipse.

I would like to connect with you more efficiently, could you please IM me on my mail id?

Thanks!

Log in to Reply

Bruno on **February 25, 2017 at 5:59 pm** said:

Yes, I think so too. Check the 2 Java versions.

Log in to Reply

kgarnu17 on **August 28, 2017 at 5:49 pm** said:

Hi Bruno,
Would you please illustrate the creation of standard agents?

Log in to Reply

Leave a Reply

You must be logged in to post a comment.