

[Home](#)
[References](#)

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 the SOAP Web Service Used in this Example
2. Creating a new SOAP Connector Integration in Pega 7
3. Calling the External SOAP Web Service from an Activity
4. Calling the External SOAP Web Service from a Data Page

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:

```

1. <soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
2.     xmlns:ser="https://services.pegaxchange.com">
3.     <soapenv:Header/>
4.     <soapenv:Body>
5.         <ser:getAllProducts/>

```

RECENT POSTS

- [Create a REST Service in Pega 7 using REST Rule](#) October 1, 2017
- [Configure Pega 7 to use HT Services](#) September 21, 2017
- [Running Pega 7 Exercise S and PostgreSQL – without VirtualBox](#) September 16, 2017
- [Add Entry to PATH Variable Sierra: for Current Terminal Permanently for Current User for All Users \(Global\)](#) June 1, 2017
- [How to Use Expressions and Functions for String Manipulation or Activity](#) June 1, 2017
- [How to Delete a Class and Pega 7](#) May 29, 2017
- [Source Dropdown Options and Group by a Property in](#)

```

6.    </soapenv:Body>
7.    </soapenv:Envelope>

```

getAllProducts response:

```

1.    <soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
2.        xmlns:xsd="https://www.w3.org/2001/XMLSchema"
3.        xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
4.        <soapenv:Body>
5.            <getAllProductsResponse xmlns="https://services.pegaxchange.com">
6.                <getAllProductsReturn>
7.                    <category>Electronics</category>
8.                    <id>1</id>
9.                    <name>Keyboard</name>
10.                   <unitPrice>29.99</unitPrice>
11.                </getAllProductsReturn>
12.                <getAllProductsReturn>
13.                    <category>Electronics</category>
14.                    <id>2</id>
15.                    <name>Mouse</name>
16.                    <unitPrice>9.95</unitPrice>
17.                </getAllProductsReturn>
18.                <getAllProductsReturn>
19.                    <category>Electronics</category>
20.                    <id>3</id>
21.                    <name>17" Monitor</name>
22.                    <unitPrice>159.49</unitPrice>
23.                </getAllProductsReturn>
24.                <getAllProductsReturn>
25.                    <category>Hardware</category>
26.                    <id>4</id>
27.                    <name>Hammer</name>
28.                    <unitPrice>9.95</unitPrice>
29.                </getAllProductsReturn>
30.                <getAllProductsReturn>
31.                    <category>Hardware</category>
32.                    <id>5</id>
33.                    <name>Slot Screwdriver</name>
34.                    <unitPrice>7.95</unitPrice>
35.                </getAllProductsReturn>
36.                <getAllProductsReturn>
37.                    <category>Books</category>
38.                    <id>6</id>
39.                    <name>The British Invasion of Java</name>
40.                    <unitPrice>11.39</unitPrice>
41.                </getAllProductsReturn>
42.                <getAllProductsReturn>
43.                    <category>Books</category>
44.                    <id>7</id>
45.                    <name>A House in Bali</name>
46.                    <unitPrice>15.99</unitPrice>
47.                </getAllProductsReturn>
48.                <getAllProductsReturn>
49.                    <category>Books</category>
50.                    <id>8</id>
51.                    <name>An Alaskan Odyssey</name>
52.                    <unitPrice>799.99</unitPrice>
53.                </getAllProductsReturn>
54.                <getAllProductsReturn>
55.                    <category>Electronics</category>
56.                    <id>9</id>
57.                    <name>LCD Projector</name>
58.                    <unitPrice>1199.19</unitPrice>
59.                </getAllProductsReturn>
60.            </getAllProductsResponse>
61.        </soapenv:Body>
62.    </soapenv:Envelope>

```

insertProduct request:

```

1.    <soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
2.        xmlns:ser="https://services.pegaxchange.com">
3.        <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 /](#)
- [Installing a Desktop GUI on System Virtual Machine with](#)
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

CATEGORIES

- [Administration](#) (9)
 - [System Debugging](#) (2)
 - [System Tuning and Main](#)
- [Application Design](#) (1)
 - [Circumstancing](#) (1)
- [Automating Business Polici](#)
 - [Activities](#) (2)
 - [Automating Decisions](#) (
 - [Data Transforms](#) (2)
- [Data Model](#) (3)
 - [Controlling Data with Pr](#)
 - [Source Your Data with C](#)
- [Data Tables](#) (3)
- [File Service](#) (1)
- [Integration](#) (9)
 - [Configuring a REST Cor](#)

```

4.    <soapenv:Body>
5.        <ser:insertProduct>
6.            <ser:product>
7.                <ser:category>Books</ser:category>
8.                <ser:id>10</ser:id>
9.                <ser:name>A History of Western Music</ser:name>
10.               <ser:unitPrice>17.95</ser:unitPrice>
11.            </ser:product>
12.        </ser:insertProduct>
13.    </soapenv:Body>
14. </soapenv:Envelope>

```

- [Configuring a SOAP Co](#)
- [E-Mail](#) (1)
- [Interacting with an Exte](#)
- [Setup a File Listener](#) (1)
- [Java EE Web Development](#)
- [macOS](#) (1)
- [Reporting](#) (1)
 - [Configuring Reports](#) (1)
 - [Creating Business User](#)
- [UI Controls](#) (3)

insertProduct response (SUCCESS):

```

1. <soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
2.     xmlns:xsd="https://www.w3.org/2001/XMLSchema"
3.     xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
4.     <soapenv:Body>
5.         <insertProductResponse xmlns="https://services.pegaxchange.com"/>
6.     </soapenv:Body>
7. </soapenv:Envelope>

```

searchById request:

```

1. <soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
2.     xmlns:ser="https://services.pegaxchange.com">
3.     <soapenv:Header/>
4.     <soapenv:Body>
5.         <ser:searchById>
6.             <ser:id>10</ser:id>
7.         </ser:searchById>
8.     </soapenv:Body>
9. </soapenv:Envelope>

```

searchById response (SUCCESS):

```

1. <soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
2.     xmlns:xsd="https://www.w3.org/2001/XMLSchema"
3.     xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
4.     <soapenv:Body>
5.         <searchByIdResponse xmlns="https://services.pegaxchange.com">
6.             <searchByIdReturn>
7.                 <category>Books</category>
8.                 <id>10</id>
9.                 <name>A History of Western Music</name>
10.                <unitPrice>17.95</unitPrice>
11.            </searchByIdReturn>
12.        </searchByIdResponse>
13.    </soapenv:Body>
14. </soapenv:Envelope>

```

searchById response (FAIL):

```

1. <soapenv:Envelope xmlns:soapenv="https://schemas.xmlsoap.org/soap/envelope/"
2.     xmlns:xsd="https://www.w3.org/2001/XMLSchema"
3.     xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance">
4.     <soapenv:Body>
5.         <soapenv:Fault>
6.             <faultcode>soapenv:Server.userException</faultcode>
7.             <faultstring>java.lang.Exception: No product found with id 11</faultstring>
8.             <detail>
9.                 <ns1:hostname xmlns:ns1="https://xml.apache.org/axis/">prpc</ns1:hostname>
10.            </detail>
11.        </soapenv:Fault>
12.    </soapenv:Body>
13. </soapenv:Envelope>

```

ARCHIVES

- [October 2017](#) (1)
- [September 2017](#) (2)
- [June 2017](#) (2)
- [May 2017](#) (3)
- [April 2017](#) (2)
- [February 2017](#) (1)
- [January 2017](#) (1)
- [October 2016](#) (1)
- [September 2016](#) (1)
- [August 2016](#) (1)
- [July 2016](#) (1)
- [June 2016](#) (1)
- [May 2016](#) (3)
- [April 2016](#) (1)
- [February 2016](#) (2)
- [January 2016](#) (5)
- [December 2015](#) (2)

- 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>



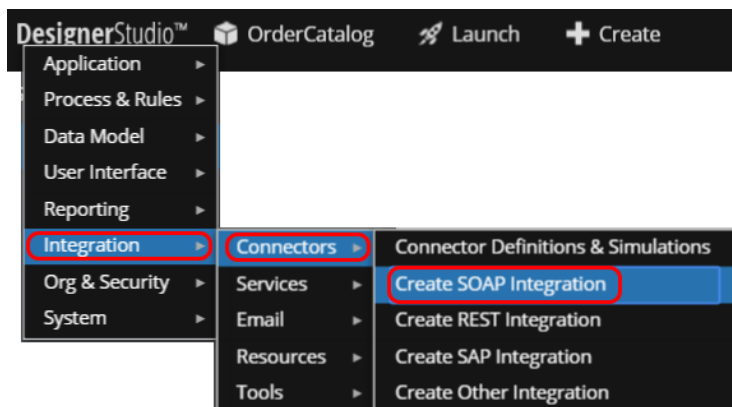
- ...and the WSDL file is accessed by adding `?wsdl` to the endpoint URL:

<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>

Home New SOAP Inte... Help

New SOAP Integration

1 Load WSDL 2 Select Operations 3 Generate Records

☒ Upload WSDL via URL
☐ Upload WSDL from File

<http://localhost:8080/ProductCatalogSOAPService/services/ProductCatalogServiceImpl?wsdl>

Next > Cancel

- 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.

Home New SOAP Inte... Help

New SOAP Integration

1 Load WSDL 2 Select Operations 3 Generate Records

Service Name
ProductCatalogServiceImplService

Port Name
ProductCatalogServiceImpl

Endpoint URL
<http://localhost:8080/ProductCatalogSOAPService/services/ProductCatalogServiceImpl> Edit

[Edit Authentication](#)

[View WSDL](#)

SELECT	OPERATION NAME	
<input checked="" type="checkbox"/>	getAllProducts	Test
<input checked="" type="checkbox"/>	searchById	Test
<input checked="" type="checkbox"/>	insertProduct	Test

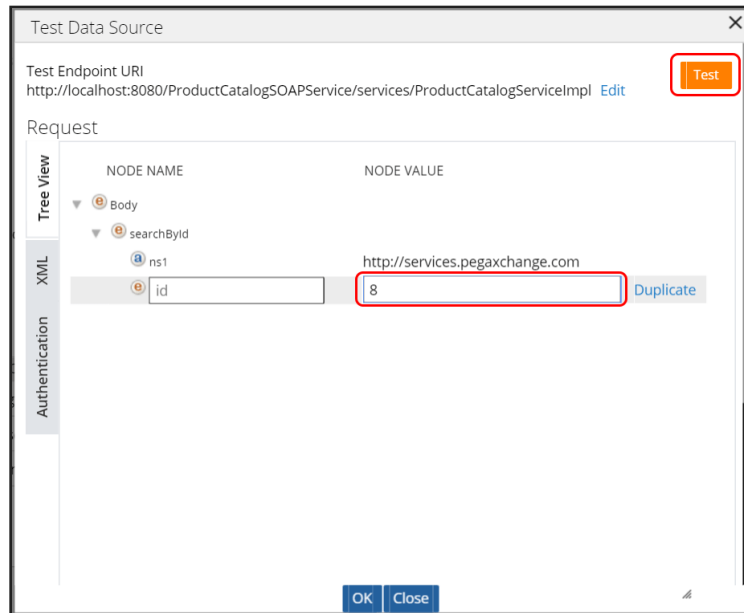
[Select All](#) [Deselect All](#)

Next > Cancel

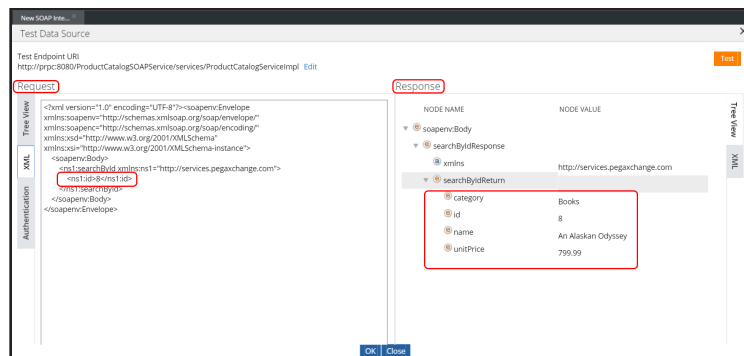
Test Data Source

- 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.

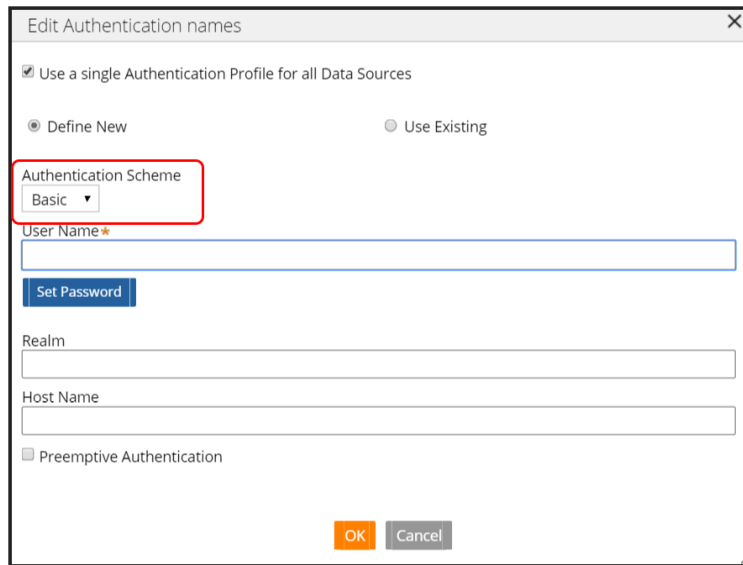
- This method has one input parameter, named **id**, to search for a product in the catalog by unique ID.



- 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.



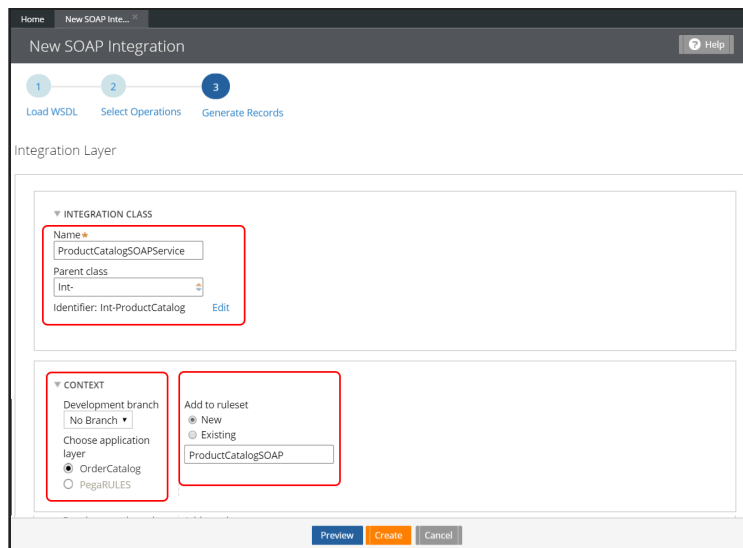
- Click on the **Ok** button at the bottom of the screen to close the **Test Data Source** dialog.
- **Optional:** The **Edit Authentication** link on the main screen of step 2 allows to configure authentication, if needed.
- See the Pega 7 help page [Create SOAP Integration](#) for more information 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.



Dialog box titled "Edit Authentication names". It contains the following elements:

- ☒ Use a single Authentication Profile for all Data Sources
- ☒ Define New ☐ Use Existing
- Authentication Scheme: Basic (highlighted with a red box)
- User Name*: (empty text field)
- Set Password: (button)
- Realm: (empty text field)
- Host Name: (empty text field)
- ☐ Preemptive Authentication
- OK and Cancel buttons at the bottom right.

- 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**.



Wizard screen titled "New SOAP Integration" showing the "Integration Layer" step. It includes a progress bar with steps 1 (Load WSDL), 2 (Select Operations), and 3 (Generate Records). The "Integration Layer" section contains:

- INTEGRATION CLASS**
 - Name*: ProductCatalogSOAPService
 - Parent class: Int-
 - Identifier: Int-ProductCatalog (with an Edit link)
- CONTEXT**
 - Development branch: No Branch
 - Choose application layer:
 - ☒ OrderCatalog
 - ☐ PegaRULES
 - Add to ruleset:
 - ☒ New
 - ☐ Existing
 - ProductCatalogSOAP

Buttons at the bottom: Preview, Create, and Cancel.

- 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.

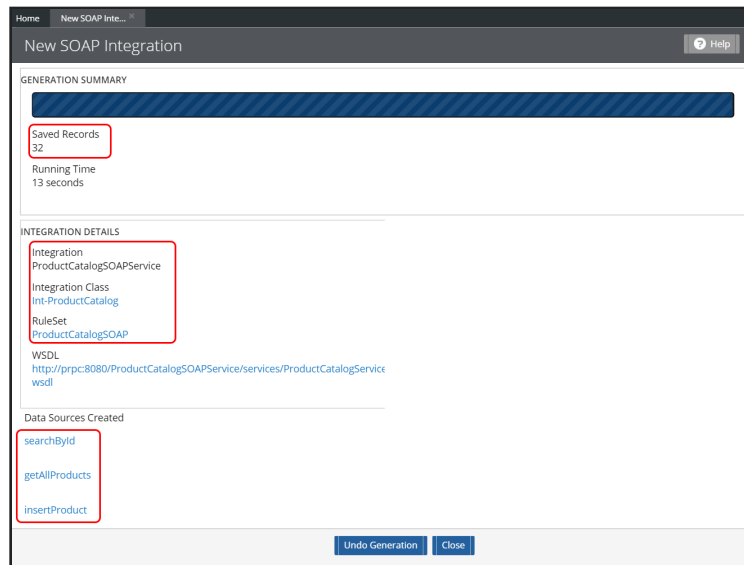


Dialog box titled "Generation Preview" showing a table of generated records:

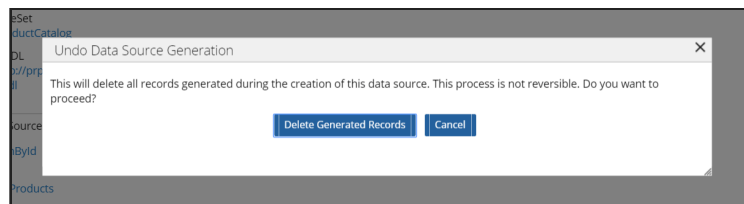
RECORD TYPE	COUNT
Class	8
Property	14
Connect SOAP	3
Parse XML	3
XML Stream	3
Authentication Profile	1
Total	32

Buttons at the bottom: OK and Cancel.

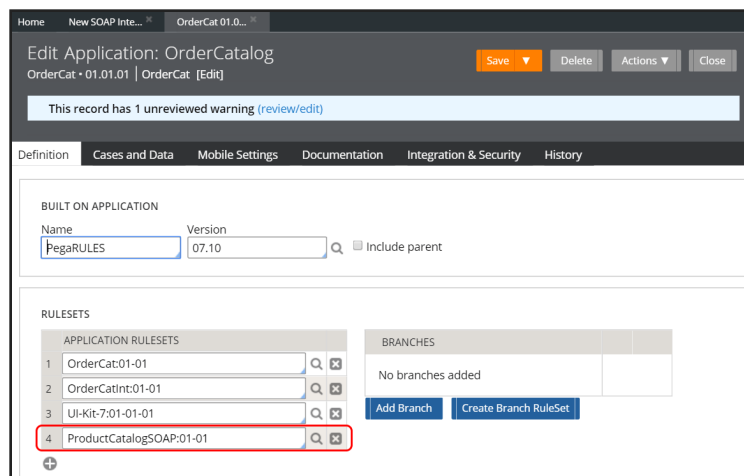
- Click on the **Create** button to start the automatic rule generation by the wizard.
- This process may take a few minutes. After completion, a confirmation screen will show a summary of the automatic rule generation.
- In the below example, 32 rules and 3 data sources (one *Connect SOAP* rule for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



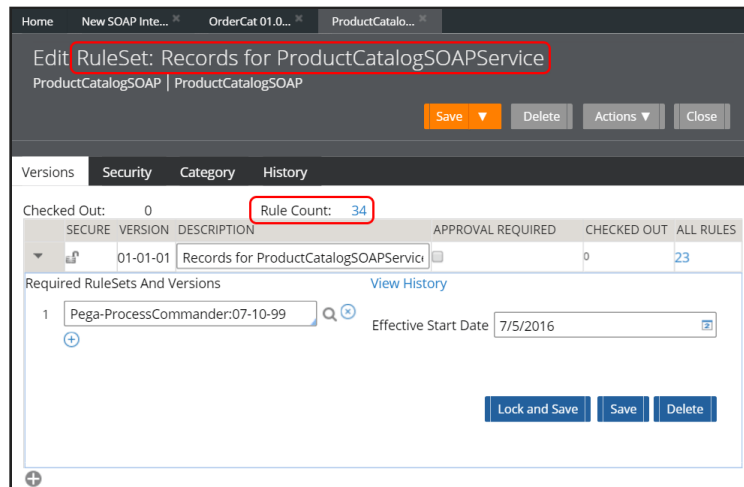
- All rules created during the generation can be removed from the system by clicking on the **Undo Generation** button.
- A confirmation dialog will appear and clicking on **Delete Generated Records** will trigger the undo process.



- 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.



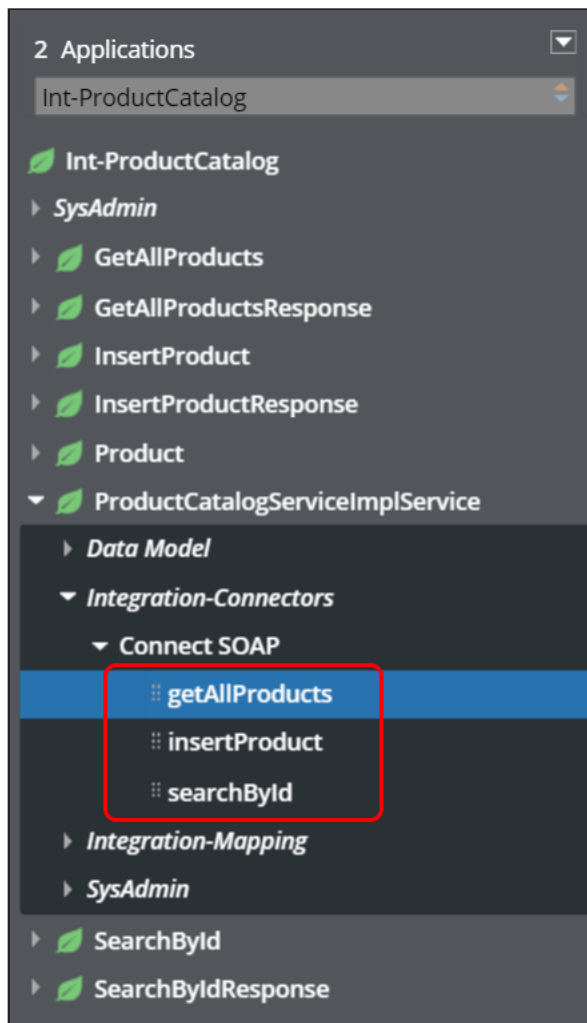
- The **Rules in RuleSet...** table view allows to drill down into specific rule types.

RULE TYPE	TOTAL COUNT
Property	14
Class	9
Connect SOAP	3
XML Stream	3
Parse XML	3
RuleSet	1
RuleSet Version	1
Grand Total	34

- 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.

RULE TYPE	RULE NAME	DESCRIPTION	APPLIES TO	AVAILABLE	CIRCUMSTANCE	UPDATED BY	LAST UPDATED
Connect SOAP	searchById		Int-Pro...	Yes		Administrator	25 minutes ago
Connect SOAP	getAllProducts		Int-Pro...	Yes		Administrator	25 minutes ago
Connect SOAP	insertProduct		Int-Pro...	Yes		Administrator	25 minutes ago

- 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.

Connect SOAP: getAllProducts (Available)
Int ProductCatalog-ProductCatalogServiceImplService • getAllProducts | ProductCatalogSOAP:01-01-01

Check out Save as Delete Actions Close

Service Request Response Faults Advanced WSDL History

SERVICE PROPERTIES

Style and use Document/literal ☒

Method Name getAllProducts

Namespace URI

SOAPAction Header

Request Only ☐

AUTHENTICATION

Use Authentication? ☐

Authentication Profile

CONNECTION

Service Endpoint URL http://prc:8080/ProductCatalogSOAPService/services/ProductC

Response Timeout 120000

Connection ID

Maintain Session ☐

ERROR HANDLING

Status Value Property pyStatusValue

Status Message Property pyStatusMessage

Invocation Exception Property pyInvocationException

Error Handler Flow ConnectionProblem

PROCESSING OPTIONS

Intended for immediate execution

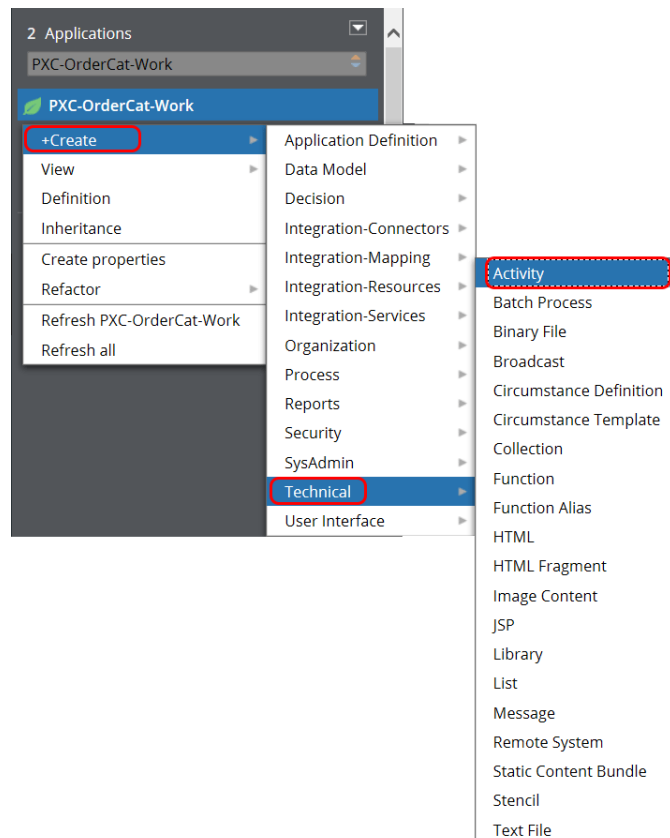
Request Processor

Number of simulations defined: 0.

Test Connectivity Simulations

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**.



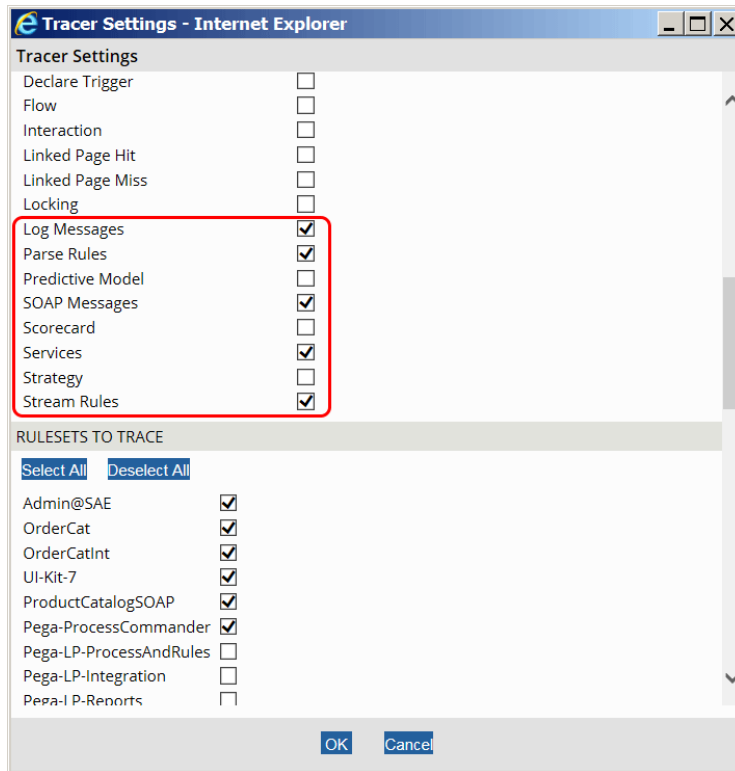
- In this case, the activity will retrieve all products from the service using the Connect SOAP rule for **getAllProducts**.
- On the **Pages and Class** tab, set a page name referring to the generated **...ImplService** class:

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:

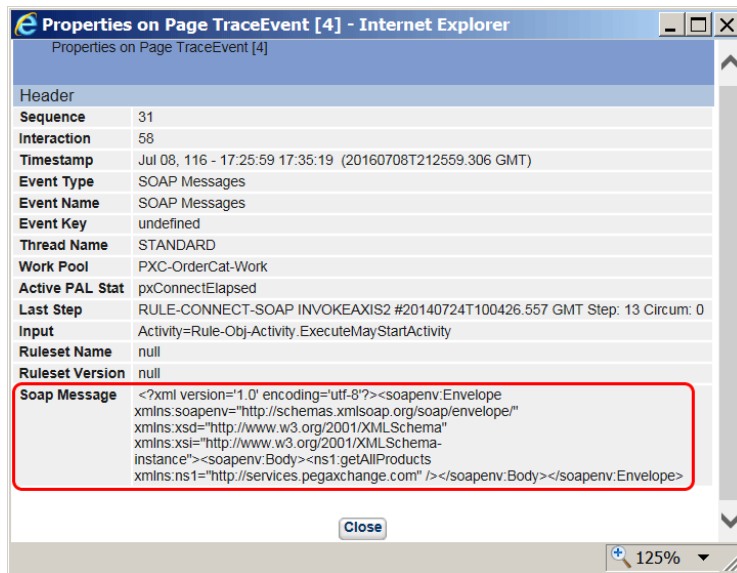
- The activity can be run by clicking on **Actions > Run**.
- Open the **Tracer** to monitor the SOAP request and response XML.
- In **Tracer Settings**, enable these items: **Log Messages**, **SOAP Messages**, **Parse Rules**, **Stream Rules** and **Services**
- Also make sure that the rulesets containing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



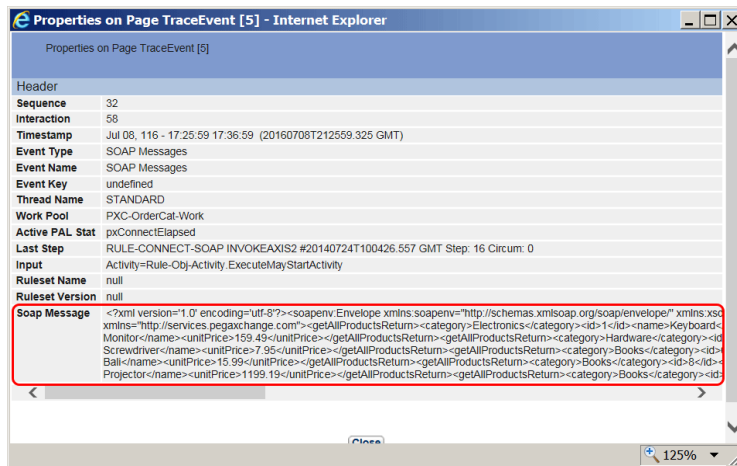
- The Tracer will show the log messages containing the product data received from the service.

LINE	THREAD	INT	RULE/STEP METHOD	STEP PAGE	STEP	STATUS	EVENT TYPE	ELAPSED	NAME	RULESET
26	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0020	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
25	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 10 Name: A History of Western Music Category: Books Price: 17.95	PXC-OrderCat-Work
24	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0010	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
23	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 9 Name: LCD Projector Category: Electronics Price: 1199.19	PXC-OrderCat-Work
22	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0010	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
21	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 8 Name: Ben & Jerry's Supreme Category: Books Price: 799.99	PXC-OrderCat-Work
20	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0010	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
19	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 7 Name: A House in Bali Category: Books Price: 15.99	PXC-OrderCat-Work
18	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0020	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
17	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 6 Name: The British Invasion of Java Category: Books Price: 11.39	PXC-OrderCat-Work
16	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0010	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
15	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 5 Name: S&S Screwdriver Category: Hardware Price: 7.95	PXC-OrderCat-Work
14	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0010	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
13	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 4 Name: Hammer Category: Hardware Price: 9.95	PXC-OrderCat-Work
12	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0020	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
11	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 3 Name: 17" LCD Monitor Category: Electronics Price: 199.99	PXC-OrderCat-Work
10	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0010	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
9	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 2 Name: Mouse Category: Electronics Price: 8.95	PXC-OrderCat-Work
8	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Step End	0.0010	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
7	STANDARD	58	4 Log Message	ProductCatalogSOAP GetAPR	3	GOOD	Message		ID: 1 Name: Keyboard Category: Electronics Price: 28.99	PXC-OrderCat-Work
6	STANDARD	58	4 Connect-SOAP	ProductCatalogSOAP	2	GOOD	Step End	0.0310	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work
5	STANDARD	58	4 SOAP Response Message	ProductCatalogSOAP	2	GOOD	SOAP Messages		ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01	ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01
4	STANDARD	58	4 SOAP Request Message	ProductCatalogSOAP	2	GOOD	SOAP Messages		ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01	ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01
3	STANDARD	58	4 Stream End	ProductCatalogSOAP	2	GOOD	Stream Rules		ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01	ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01
2	STANDARD	58	4 Stream Begin	ProductCatalogSOAP	2	GOOD	Stream Rules		ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01	ID: ProductCatalog-ProductCatalogGet ProductCatalogSOAP 01-01-01
1	STANDARD	58	4 Page-New	ProductCatalogSOAP	1	GOOD	Step End	0.0000	PXC-OrderCat-Work GetAPR(ProductCatalogSOAP-OrderCat 01-01-01)	PXC-OrderCat-Work

- 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.

Properties on Page ProductCatalogSOAP - Internet Explorer

Properties on Page TraceEvent [ProductCatalogSOAP]

Properties on Page ProductCatalogSOAP

Name	Value																																																						
pyStatusValue	Good																																																						
pxObjClass	Int-ProductCatalog-ProductCatalogServiceImpiService																																																						
pyHTTPResponseCode	200																																																						
pyInvocationException																																																							
pzStatus	valid																																																						
GetAllProductsParameters	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>pxObjClass</td><td>Int-ProductCatalog-GetAllProducts</td></tr></tbody></table>	Name	Value	pxObjClass	Int-ProductCatalog-GetAllProducts																																																		
Name	Value																																																						
pxObjClass	Int-ProductCatalog-GetAllProducts																																																						
GetAllProductsParametersResponse	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>pxObjClass</td><td>Int-ProductCatalog-GetAllProductsResponse</td></tr><tr><td>getAllProductsReturn(1)</td><td><table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>1</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>Keyboard</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>29.99</td></tr></tbody></table></td></tr><tr><td>getAllProductsReturn(2)</td><td><table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>2</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>Mouse</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>9.95</td></tr></tbody></table></td></tr><tr><td>getAllProductsReturn(3)</td><td><table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>3</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>17" Monitor</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>159.49</td></tr></tbody></table></td></tr><tr><td>getAllProductsReturn(4)</td><td><table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>4</td></tr><tr><td>category</td><td>Hardware</td></tr></tbody></table></td></tr></tbody></table>	Name	Value	pxObjClass	Int-ProductCatalog-GetAllProductsResponse	getAllProductsReturn(1)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>1</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>Keyboard</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>29.99</td></tr></tbody></table>	Name	Value	id	1	category	Electronics	name	Keyboard	pxObjClass	Int-ProductCatalog-Product	unitPrice	29.99	getAllProductsReturn(2)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>2</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>Mouse</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>9.95</td></tr></tbody></table>	Name	Value	id	2	category	Electronics	name	Mouse	pxObjClass	Int-ProductCatalog-Product	unitPrice	9.95	getAllProductsReturn(3)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>3</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>17" Monitor</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>159.49</td></tr></tbody></table>	Name	Value	id	3	category	Electronics	name	17" Monitor	pxObjClass	Int-ProductCatalog-Product	unitPrice	159.49	getAllProductsReturn(4)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>4</td></tr><tr><td>category</td><td>Hardware</td></tr></tbody></table>	Name	Value	id	4	category	Hardware
Name	Value																																																						
pxObjClass	Int-ProductCatalog-GetAllProductsResponse																																																						
getAllProductsReturn(1)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>1</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>Keyboard</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>29.99</td></tr></tbody></table>	Name	Value	id	1	category	Electronics	name	Keyboard	pxObjClass	Int-ProductCatalog-Product	unitPrice	29.99																																										
Name	Value																																																						
id	1																																																						
category	Electronics																																																						
name	Keyboard																																																						
pxObjClass	Int-ProductCatalog-Product																																																						
unitPrice	29.99																																																						
getAllProductsReturn(2)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>2</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>Mouse</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>9.95</td></tr></tbody></table>	Name	Value	id	2	category	Electronics	name	Mouse	pxObjClass	Int-ProductCatalog-Product	unitPrice	9.95																																										
Name	Value																																																						
id	2																																																						
category	Electronics																																																						
name	Mouse																																																						
pxObjClass	Int-ProductCatalog-Product																																																						
unitPrice	9.95																																																						
getAllProductsReturn(3)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>3</td></tr><tr><td>category</td><td>Electronics</td></tr><tr><td>name</td><td>17" Monitor</td></tr><tr><td>pxObjClass</td><td>Int-ProductCatalog-Product</td></tr><tr><td>unitPrice</td><td>159.49</td></tr></tbody></table>	Name	Value	id	3	category	Electronics	name	17" Monitor	pxObjClass	Int-ProductCatalog-Product	unitPrice	159.49																																										
Name	Value																																																						
id	3																																																						
category	Electronics																																																						
name	17" Monitor																																																						
pxObjClass	Int-ProductCatalog-Product																																																						
unitPrice	159.49																																																						
getAllProductsReturn(4)	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>id</td><td>4</td></tr><tr><td>category</td><td>Hardware</td></tr></tbody></table>	Name	Value	id	4	category	Hardware																																																
Name	Value																																																						
id	4																																																						
category	Hardware																																																						

Close

125%

- 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`.

Home InsertProduct...

Edit Activity: InsertProductIntoCatalog (Available)

PXC-OrderCat Work - InsertProductIntoCatalog | OrderCat:01-01-01

Save Delete Actions Close

Steps Parameters Pages and Classes Security History

1. Label Loop When Method Step Page Description

ProductCatalogSOAP Create a page

Jump

2. Label Loop When Method Step Page Description

AP:InsertProductParameters Set property values

Jump

3. Label Loop When Method Step Page Description

ProductCatalogSOAP Call the product catalog SOAP service

Jump

METHOD PARAMETERS

PROPERTYNAME PROPERTYVALUE

id 10

category Electronics

name Fernseher

unitPrice 479.85

METHOD PARAMETERS

ServiceName insertProduct

ExecutionMode Run

EndPointURL

Add a step Collapse all steps

- 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.

Home SearchProduct... Edit Activity: SearchProductByd (Available) PXC-OrderCat-Work • SearchProductByd | OrderCat:01-01-01

Steps Parameters Pages and Classes Security History

1. Loop When Page-New ProductCatalogSOAP Create a page Jump

2. Loop When Property-Set ProductCatalogSOAP-Search Set property values Jump

METHOD PARAMETERS

PROPERTIESNAME id PROPERTIESVALUE 8

3. Loop When Connect-SOAP ProductCatalogSOAP Call the product catalog SOAP service Jump

METHOD PARAMETERS

ServiceName searchByd ExecutionMode Run EndPointURL

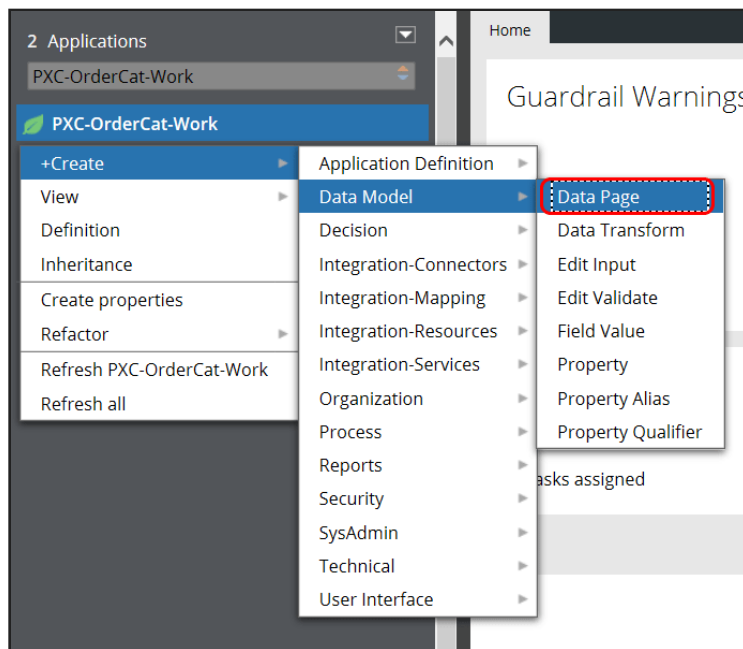
4. Loop When Log-Message response.searchBydReturn Log a message Jump

METHOD PARAMETERS

Message "ID: " + id + "Name: " + name + "Category: " + category + "Price: " + unitPrice LoggingLevel Info GenerateStackTrace SendToTracer

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**.

Home New

Create Data Page Create and open Cancel ?

DATA PAGE RECORD CONFIGURATION

Label* Identifier

A short description or title for this record

CONTEXT

Choose app layer

☒ OrderCatalog ☐ PegaRULES

Apply to View all

Add to ruleset*

UI Inspector UI Tree Performance Alerts Inspection Prefs Pega 7.1.6

- 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)

Home New

Edit Data Page: D_AllCatalogProducts (Available) Save Actions Close

D_AllCatalogProducts | OrderCat:01-01-01

Definition Load Management Parameters Pages & Classes Usage History

DATA PAGE DEFINITION

Structure

Object Type*

Edit Mode

Scope

Requestor

KEYED PAGE ACCESS

☐ Access pages with user defined keys

DATA SOURCES

☐ Simulate Data Source

Source*

Type Name*

Int-ProductCatalog...

Request Data Transform Response Page List

Response Data Transform*

Endpoint URL

Create Data Source

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

Home D_AllCatalogP... GetProductsFrom...

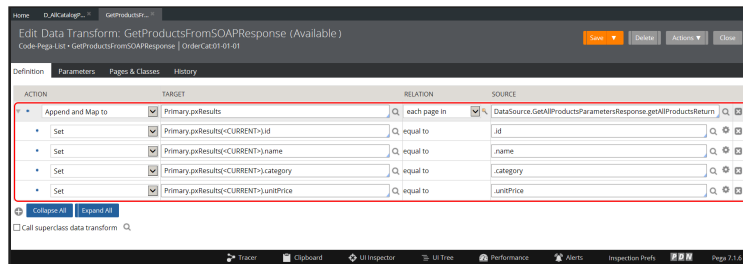
Edit Data Transform: GetProductsFromSOAPResponse (Available) Save Delete Actions Close

Code-Pega-List - GetProductsFromSOAPResponse | OrderCat:01-01-01

Definition Parameters Pages & Classes History

PAGE NAME	CLASS
DataSource	Int-ProductCatalog-ProductCatalogServiceImplService
Primary.pxResults	Int-ProductCatalog-Product

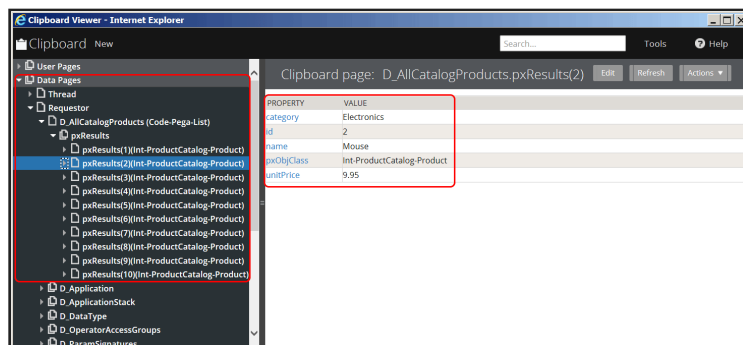
- The **Definition** tab, linking the service response to the data page list structure looks like this (... [click on image to enlarge](#)):



- 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](#)

rahulID 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](#)

rahulID 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.pegaxchange.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.

