VS Process Designer

Tutorial 5: Importing a Service

CONTENTS

This tutorial assumes that you have completed the previous tutorials and builds upon them.

In previous tutorials you used the Process Design Studio to create everything your project needed. For example, you created complex types for input and output variables. Creating a complex type involves defining a schema, or structure. In tutorial four you created an output type that had four elements, one each for the sum of nodes, the average, the product, and the sum of every other node. In addition to defining schema within the Process Design Studio, you can also import schema from an external source.

In this tutorial, you will import an external service and use schema from it as the input type in your project. The project will send a last name, state, and middle initial to a Web service, which will respond with all matching records. You will test the service as both a SOAP web service and a RESTful service. The project will then output a count of records returned.

Prerequisites:

- Micro Focus Verastream Process Design Studio
- An installed and running Micro Focus Verastream Process Server
- Internet browser
- Some familiarity with XML Schema, WSDL, XPath, BPEL, REST, JSON, and Web service standards

Let’s get started.

You can import a service at any time by selecting **File** *>* **Import Service***. You will see the Import Service dialog, shown in step 6 of this lesson.*

By default, the process server's name is 'local-host'. If your server has a different name, replace 'localhost' with the name of your server.

The service you will import, CCSDemoService, is part of the default installation for Verastream Process Designer. If you modified the default installation, you may need to reinstall or change the URL required in this process in order to complete the tutorial.

1. Start the Process Design Studio (**Start** > **Micro Focus Verastream** > **Process Designer** > **Process Design Studio**).

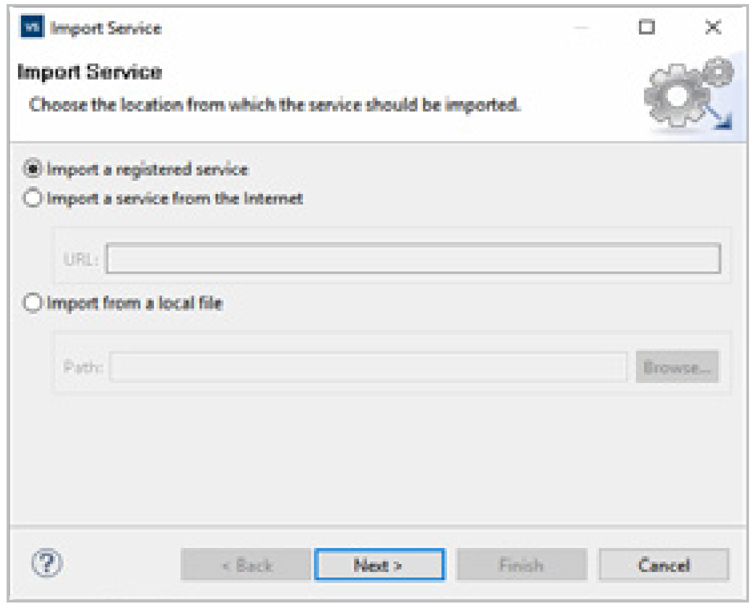
2. From the File menu, select **New Project**.

3. Name the new project **ExternalService** and click **Next**.

4. From the File menu, open the Import Services dialog box.

5. On Import Service, make sure that **Import a registered service** is selected.

Registered services are those hosted by Verastream Host Integrator or Verastream Process Designer.



Import Service

Import Service

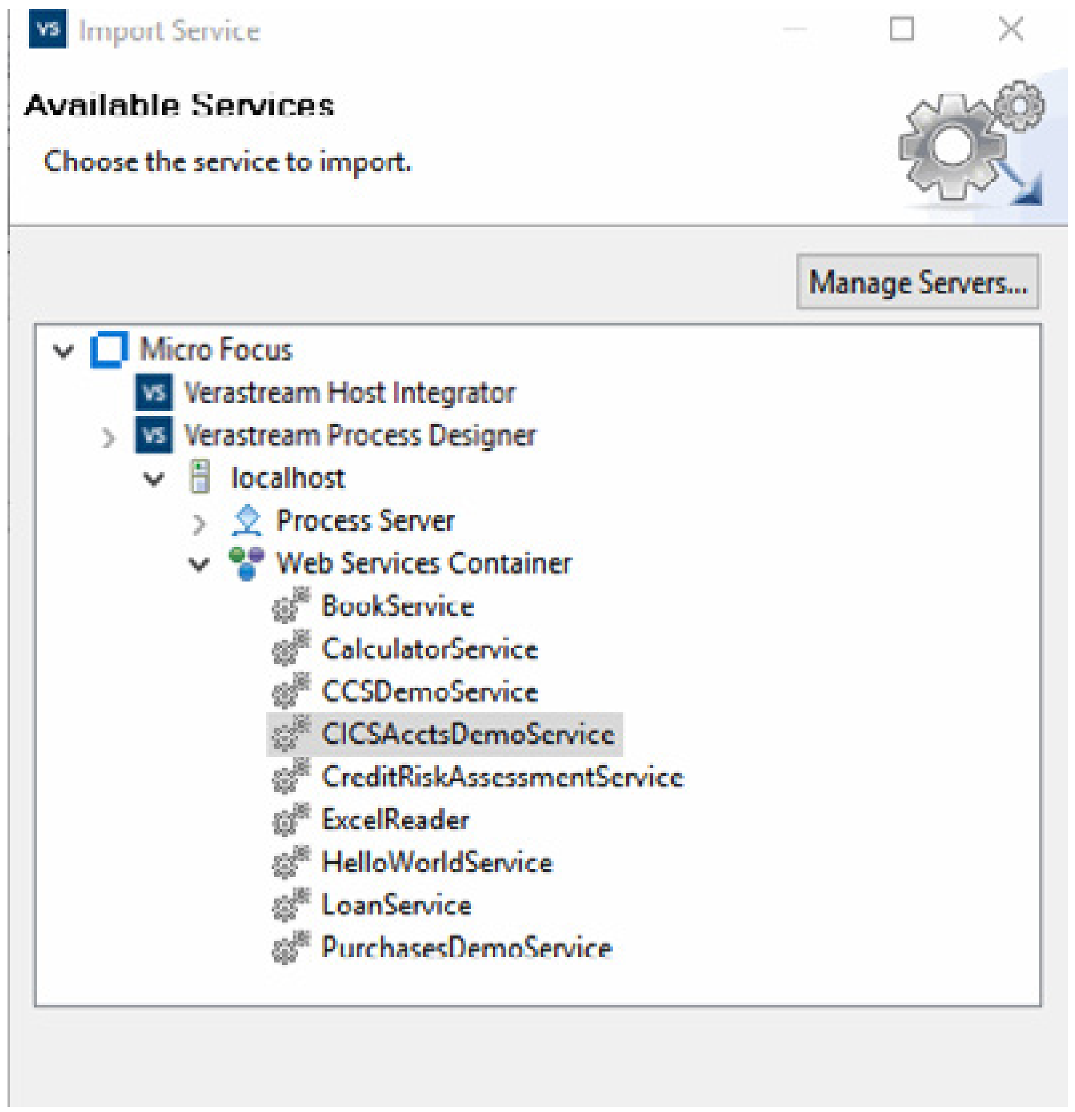
Choose the location from which the service should be imported.

[✓ Import a registered service]
Import a service from the Internet
URL:
Import from a local file
Path:

6. Click **Next**.

7. From the Available Services dialog box, expand Verastream Process Designer. If you cannot expand the node, click **Manage Servers** to add the Process Server. Add the name of the machine where the Process Server is located. In our case it is **localhost.** Click **OK.**

8. Expand localhost, Web Services Container, and then select **CCSDemoService** and click **Next**.



Import Service

Available Services

Choose the service to import.

- Micro Focus
  - Verastream Host Integrator
  - Verastream Process Designer
    - localhost
      - Process Server
      - Web Services Container
        - BookService
        - CalculatorService
        - CCSDemoService
        - CICSAcctsDemoService
        - CreditRiskAssessmentService
        - ExcelReader
        - HelloWorldService
        - LoanService
        - PurchasesDemoService

9. On the Name and Confirm dialog, the Name field is populated with the name of the service. You can change this name. Click **Finish**, then **OK**.

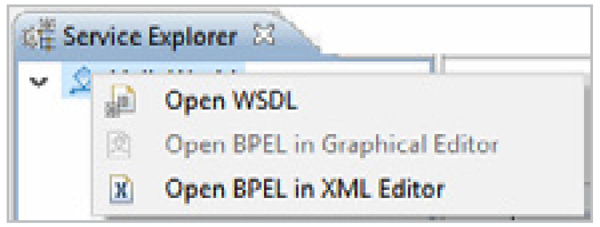
10. In the BPEL Editor, delete the DoSomethingHere activity (and the AssignValue it contains) from your default project.

11. Save the project.

Defining input and output types

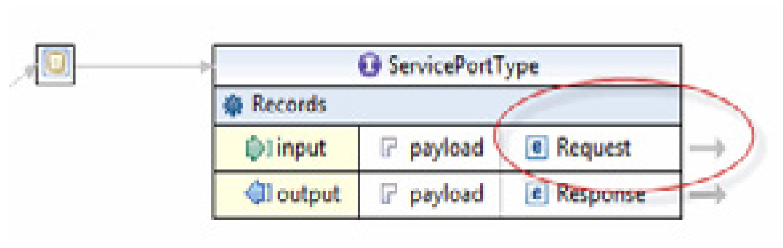
The next step is to change the input type in your project to a new type, defined in the service you just imported.

1. In the Service Explorer, right-click the top node and select **Open WSDL**. This opens the WSDL Editor. Verify you are in Design mode by checking the tabs at the bottom on the editor.



2. In the WSDL Editor, click the arrow to the right of Request to open the Schema Editor.

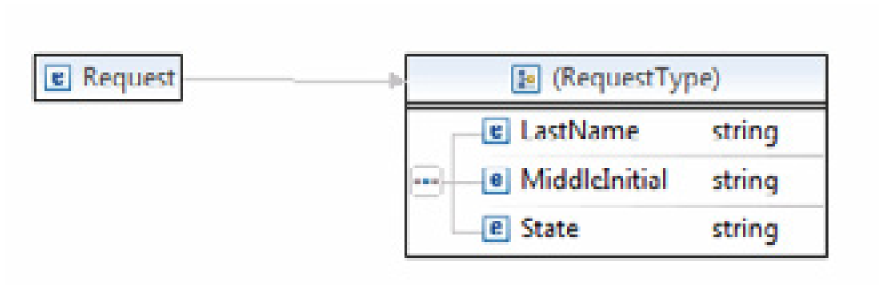
3. To add input elements to the process, right-click



RequestType and select Add Element. Repeat this step until you have three elements. You can use Input as your first element.

Types defined in imported services are added to the list of types available in your Workspace.

4. Rename the elements: LastName, MiddleInitial, and State using the General tab of the Properties view.



5. Save the project. Close the Schema Editor and the WSDL Editor.

Invoking the external service

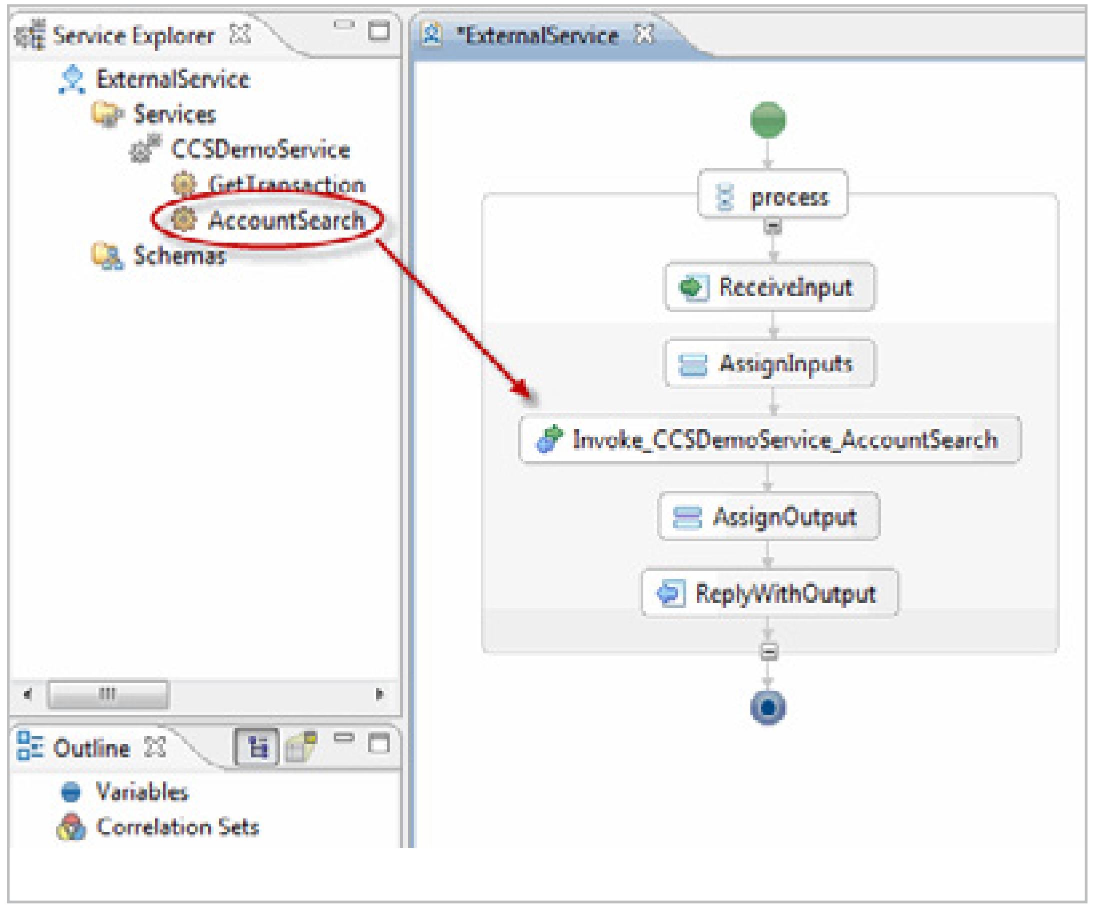
On the Preferences menu, the BPEL Property, Initialize Variables Automatically, is selected by default. This means that all variables are initialized when they are created.

To use the CCSDemo AccountSearch service, you must invoke it as part of your BPEL process. Invoking it will add its Input and Output variables to your project.

1. Insert an Assign activity between ReceiveInput and ReplyWithOutput by selecting Assign in the palette and then clicking between those activities.

2. In the Properties view, open the Description tab and change the name from Assign to **AssignInputs**.

3. Open the Service Explorer tab, expand **CCSDemoService**, and drag **AccountSearch** into the BPEL Editor. Place it between AssignInputs and ReplyWithOutput.



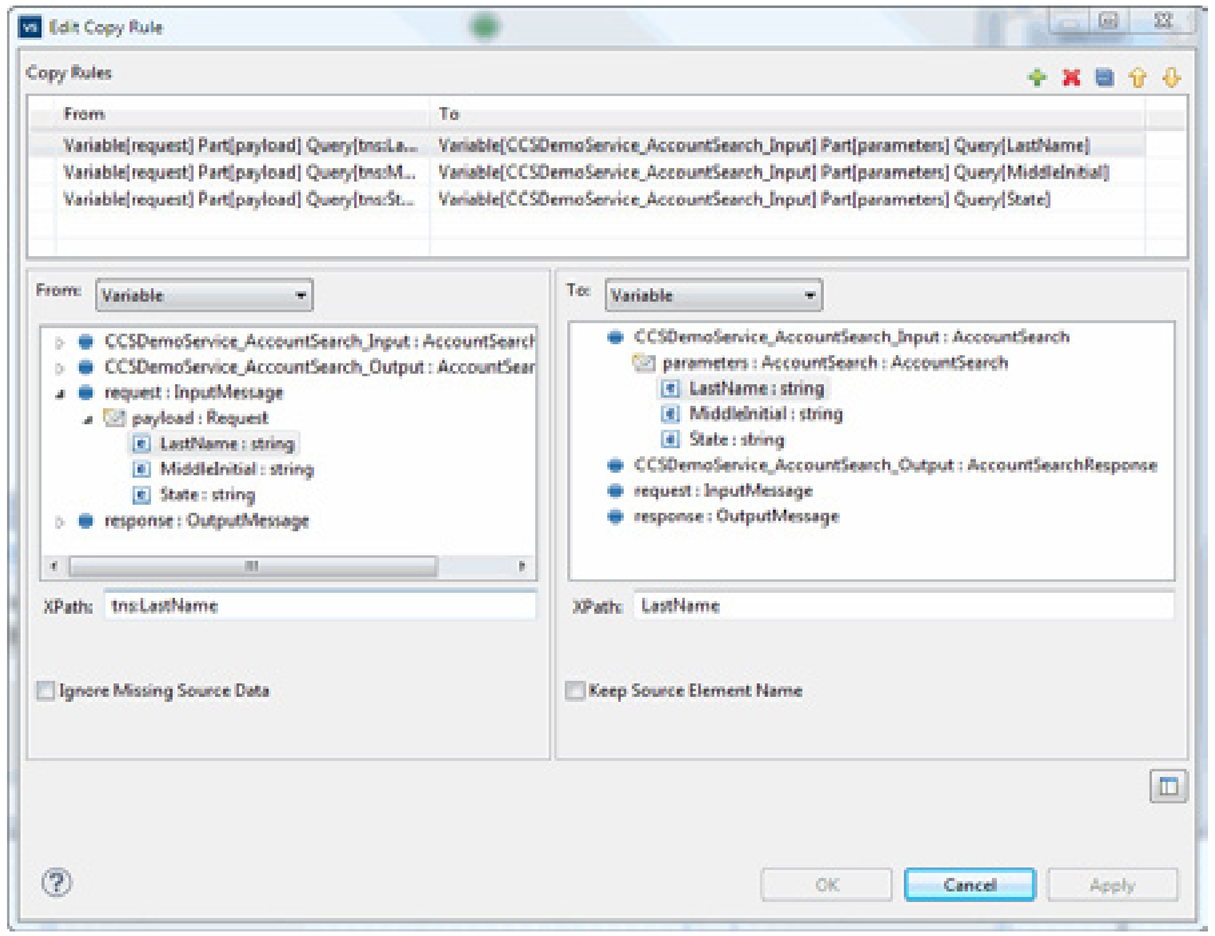
Copying input values

Your ExternalService process now receives three elements of input (last name, middle initial, state), and then invokes an external service. The next step is to copy the input coming into your process to the input variable for the external service. This must be done before the external service is invoked.

You can use the up and down arrows in the Details tab on the far right to change their order.

1. In the BPEL Editor, select **AssignInputs**, and in the Details tab of the Properties view, open the Copy Rule dialog box(✚). You are going to add three copy rules.

2. On the From side, expand `request:InputMessage` and `payload:Request, and select LastName.` On the To side, expand `CCSDemo_AccountSearch_ Input:AccountSearch`, and select `parameters:AccountS earch:AccountSearch` and select `LastName`. Repeat this process for MiddleInitial and State. You should have three copy rules. Click **OK**.


Edit Copy Rule

Copy Rules

From	To
Variable[request] Part[payload] Query[tns:La...	Variable[CCSDemoService_AccountSearch_Input] Part[parameters] Query[LastName]
Variable[request] Part[payload] Query[tns:M...	Variable[CCSDemoService_AccountSearch_Input] Part[parameters] Query[MiddleInitial]
Variable[request] Part[payload] Query[tns:St...	Variable[CCSDemoService_AccountSearch_Input] Part[parameters] Query[State]

From: Variable

- CCSDemoService_AccountSearch_Input : AccountSearch
- CCSDemoService_AccountSearch_Output : AccountSear
- request : InputMessage
  - payload : Request
    - LastName : string
    - MiddleInitial : string
    - State : string
- response : OutputMessage

XPath: tns:LastName

☐ Ignore Missing Source Data

To: Variable

- CCSDemoService_AccountSearch_Input : AccountSearch
  - parameters : AccountSearch : AccountSearch
    - LastName : string
    - MiddleInitial : string
    - State : string
- CCSDemoService_AccountSearch_Output : AccountSearchResponse
- request : InputMessage
- response : OutputMessage

XPath: LastName

☐ Keep Source Element Name

3. Save the project.

Getting and copying the number of records returned

The CCSDemo service will return a number of records in response to the input your process provides. The last step is to fill the Output variable of your process with the count of records returned by the service.

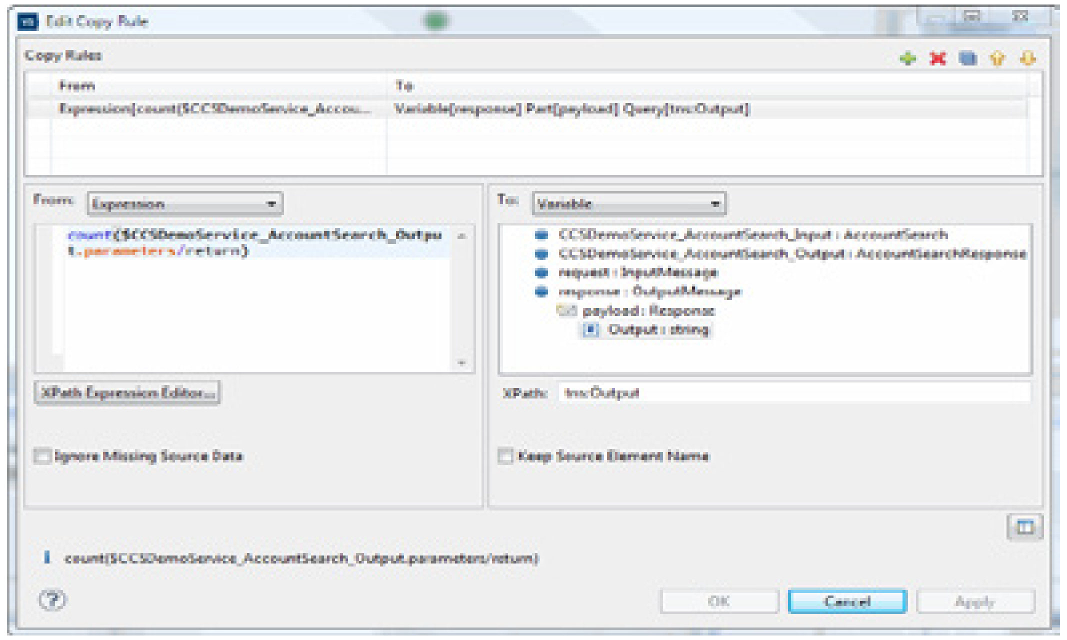
1. On the Palette, select the Assign activity, then click between Invoke_CCSDemo_AccountSearch and ReplyWithOutput.

2. Name the new Assign activity AssignOutput.

3. Select **AssignOutput**, and in the Details tab of the Properties view, add a copy rule (✚).

4. In the From dropdown, select **Expression**, then click **XPath Expression Editor...**

5. In the Functions tree, expand **Node** and double-click **count**. With `item_sequence` selected, in the Variables tree, expand `CCSDemoService_AccountSearch_ Output:AccountSearchResponse`, then expand `parameters:AccountSearchRespo nse: AccountSearchResponse` and double-click `return:accountRecord`.


Edit Copy Rule

Copy Rules

From	To
Expression[count($CCSDemoService_Accou...	Variable[response] Part[payload] Query[tns:Output]

From: Expression

```
count($CCSDemoService_AccountSearch_Outpu
t.parameters/return)
```

XPath Expression Editor...

☐ Ignore Missing Source Data

To: Variable

- CCSDemoService_AccountSearch_Input : AccountSearch
- CCSDemoService_AccountSearch_Output : AccountSearchResponse
- request : InputMessage
- response : OutputMessage
  - payload : Response
    - Output : string

XPath: tns:Output

☐ Keep Source Element Name

count($CCSDemoService_AccountSearch_Output.parameters/return)

6. Delete `[1]` (the 1 and the brackets around it). Click **OK**.

7. On the To side, expand `response:OutputMessage`, and expand `payload:Response`, and select `Output:String`. Click **OK**.

Deploying and testing as a SOAP service

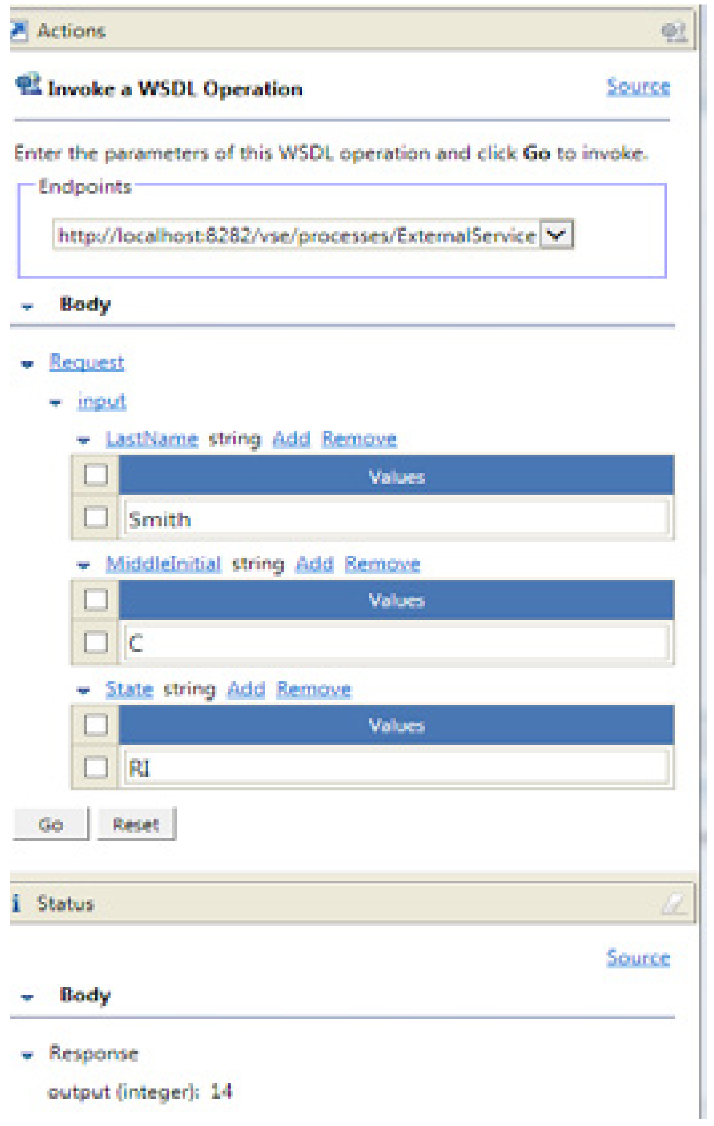
It’s time to deploy and test your new process.

1. From the File menu, select **Deploy to Process Server**.
2. Enter the name, username and password for the server.
3. In the Deployment Succeeded dialog box, click **Test Service** to launch the Web Services Explorer.
4. You will see three inputs: LastName, MiddleInitial, and State. Click the Add link next to each.
5. For LastName input **Smith**
   for MiddleInitial input **C**
   for State input **RI**
   then press **Go**.
6. The Output string should be **14**.

The default username and passwords for the local-host server are: admin and secret.

The Web Services Explorer is a browser-based tool that provides a medium to test SOAP-based Web services.

Remember to select the SOAP11Binding to test against.



Enter the parameters of this WSDL operation and click **Go** to invoke.

Endpoints: http://localhost:8282/vse/processes/ExternalService

LastName string Add Remove
Values: Smith

MiddleInitial string Add Remove
Values: C

State string Add Remove
Values: RI

output (integer): 14

Testing the process as a RESTful service

For a Verastream Process Designer service to be called RESTfully, its input must be a complex type. You cannot use simple inputs, such as strings, as input. In VPD all inputs and outputs are complex by default.

After you create and deploy your project, it is available to you as either a SOAP-based web service or a RESTful service. This allows you to execute your service using standard HTTP calls sending and receiving common data formats such as JSON and XML.

- The complete URL to the RESTful service is the base URL plus the project name and the name of the operation you want to call. For example:

  Base URL: `http://localhost:8282/vse/processes/`

  Complete URL: `http://localhost:8282/vse/processes/ExternalService/Records`

- VPD RESTful services only support the HTTP POST request type.
- Content type may be application/JSON or application/XML.

To view operations exposed by the project in the Service Explorer, you must first enable this option in Windows > Preferences > Project.

An example of using the Curl utility to call the RESTful service:

```
curl -X POST http://localhost:8282/vse/processes/
ExternalService/Records -H “Content-Type: appli-
cation/json” -d ‘{“Request”:{“LastName”:”Smith”,
“MiddleInitial”:”C”, “State”:”RI”}}’
```

If you are running Curl on Windows you may need to escape the double quotation marks.

An example of a simple JQuery request using AJAX:

```
var ajaxRequest = $.ajax({

      url: ‘http://localhost:8282/vse/processes/
ExternalService/Records’,

      contentType: ‘application/json; charset=UTF-8’,

      type: ‘POST’,

      dataType:‘json’,

      data: ‘{“Input”: {“LastName”:”Smith”,
“MiddleInitial”:”C”, “StateAbbr”:”RI”}}’

 });
```

In this example the contentType was ‘application/json’. This means that the service will be expecting the input to be in JSON format and the output will also be in JSON format:

```
{

   “Response”: {

      “Output”: 14

   }

}
```

Determining the inputs to your RESTful service

There are many free online conversion tools available on the Internet.

Verify that your inputs and outputs are compatible with JSON. Not all XML maps successfully to JSON.

You can use the Web Service Explorer and online XML to JSON conversion tools to help determine what the JSON input to your RESTful service will look like.

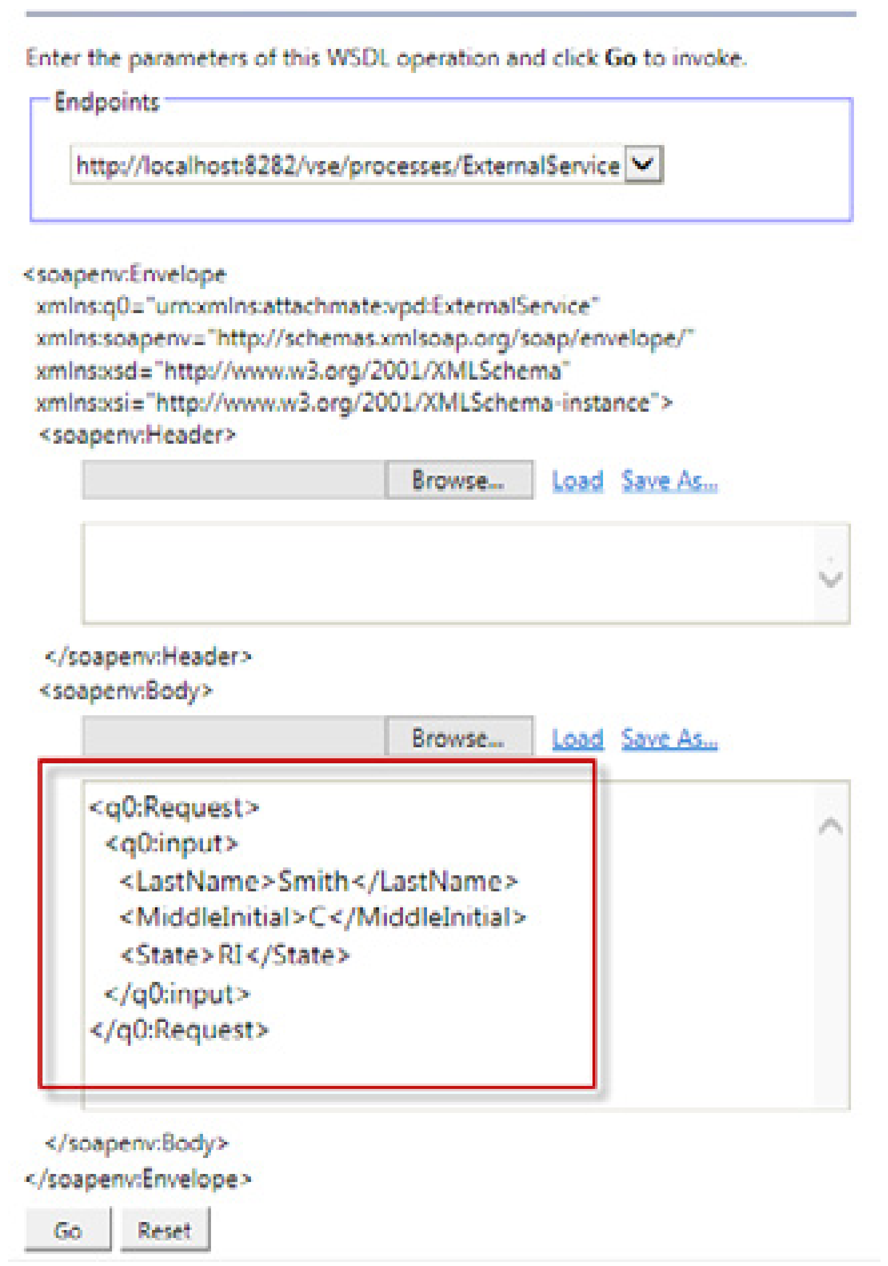
For example the operation, **Records**, takes three string inputs: **LastName, MiddleInitial**, and **State**.

The XML, generated by Web Services Explorer, looks like this:

VPD RESTful services support the POST HTTP request method. The content type may be application/XML or application/JSON.

JSON is typically used in RESTful services.

POST request bodies are used to input data to the service.


Enter the parameters of this WSDL operation and click Go to invoke.
Endpoints
http://localhost:8282/vse/processes/ExternalService
Browse...
Load
Save As...
Go
Reset

```
<soapenv:Envelope
 xmlns:q0="urn:xmlns:attachmate:vpd:ExternalService"
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Header>
 </soapenv:Header>
 <soapenv:Body>
  <q0:Request>
   <q0:input>
    <LastName>Smith</LastName>
    <MiddleInitial>C</MiddleInitial>
    <State>RI</State>
   </q0:input>
  </q0:Request>
 </soapenv:Body>
</soapenv:Envelope>
```

q0 is the name prefix of the service. When converting, do not include the Envelope or the Header.

To convert the XML to JSON:

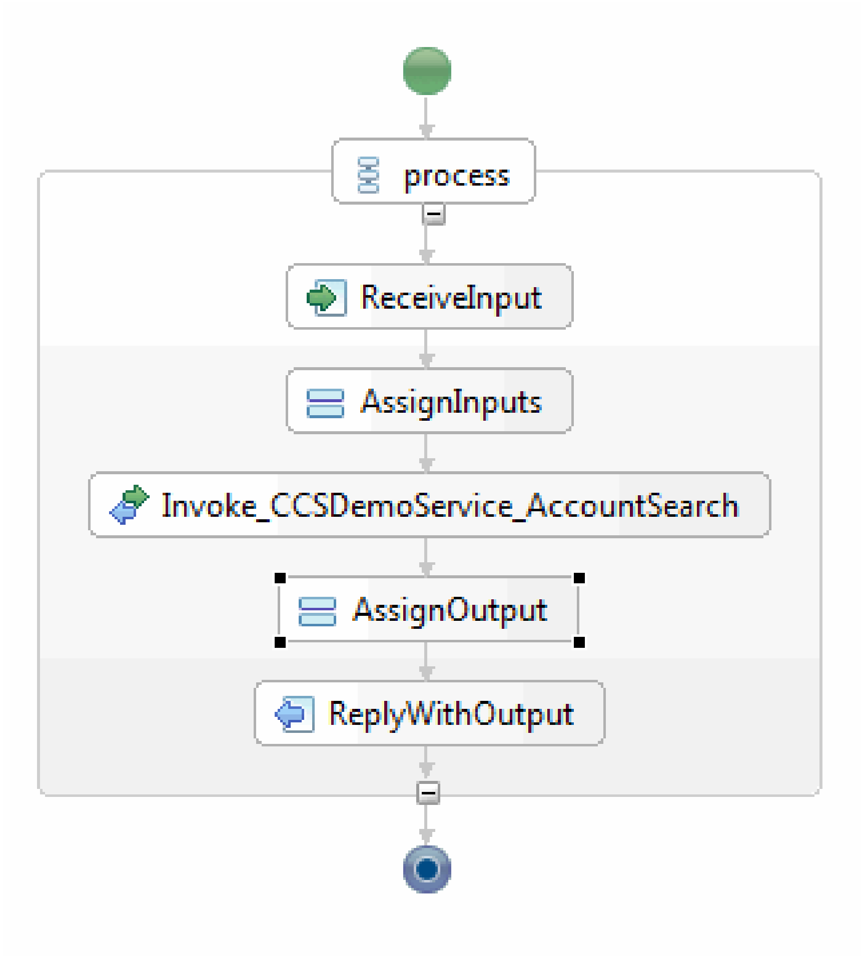
1. Remove the name prefix (q0). JSON does not use namespaces.

2. Paste the cleaned-up XML into the XML to JSON conversion tool you have selected. The JSON output should look like this:

```
{
  "Request": {
    "LastName": "Smith",
    "MiddleInitial": "C",
    "State": "RI"
  }
}
```

This will be used as the JSON input to your service.

Reference image of the completed process


process
ReceiveInput
AssignInputs
Invoke_CCSDemoService_AccountSearch
AssignOutput
ReplyWithOutput