Web Service Interface Lesson 2 MD18

All "public" Web services, which clients may access, must have a port type definition. A port type accumulates Web service operations that a client can call. In the UML model, port types will be designed by using a class stereotyped as port type. This special kind of class is called **port type** and has no attributes but only operations. Each operation must be assigned to an activity diagram of the UML model. Operations of a port type represent the interfaces of a Web service. Activities implement the behavior of these operations.

Within a Web service, one or more services can be included. Each service can have one or more port types. Within a port type, one or more operations having input and/or output parameters can be defined. Port types are the SOAP interface to the outside world.

In the next development step, you will create the SOAP interface of the Web service. In lesson 1, the SOAP Interface structure was already predefined via the E2E model template you used. Therefore, in order to learn how to create a new SOAP interface from scratch, you will not copy or rework the interface of lesson 1.
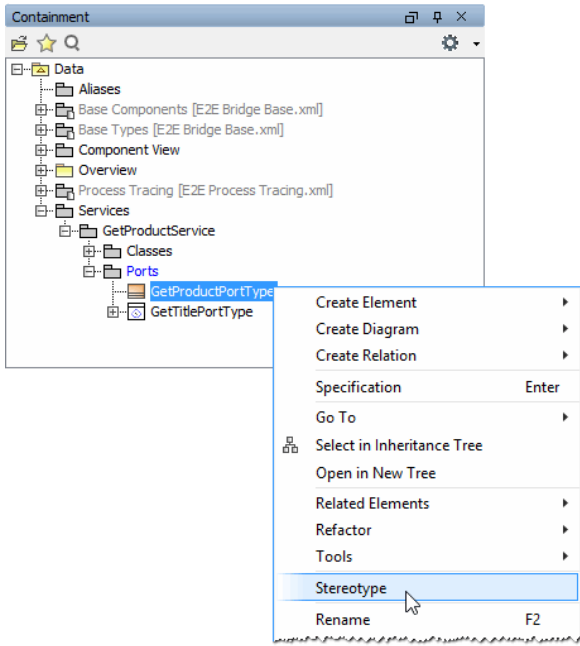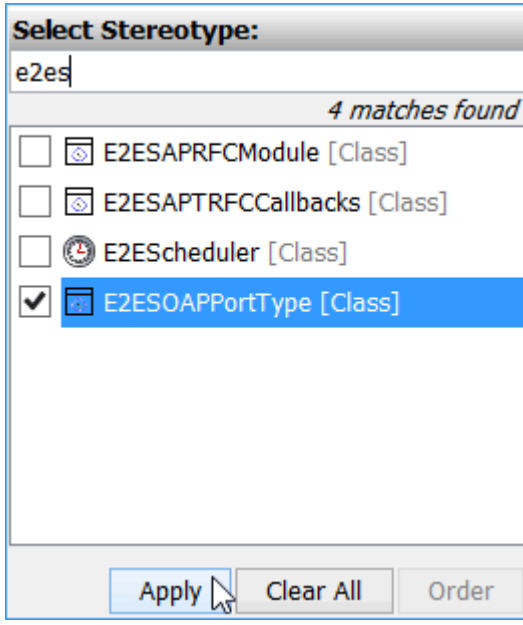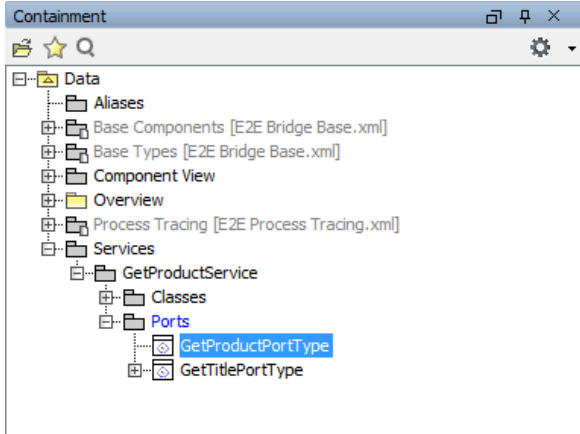
Defining the Port Type

In the containment tree, navigate to the package **Data / Services / GetProductService / Ports**, and create a new class.

Name the port type **GetProductPortType**.

Activities

On this Page:

- Defining the Port Type
- Defining the Port Type Operation
- Defining the Operation Parameters
- Assigning a New Activity Diagram to the Operation

|	

Containment ⬒ 📌 ✕
📂 ⭐ 🔍   ⚙ ▾
⊟ 🔺 **Data**
  📁 Aliases
  ⊞ 📁 Base Components [E2E Bridge Base.xml]
  ⊞ 📁 Base Types [E2E Bridge Base.xml]
  ⊞ 📁 Component View
  ⊞ 📁 Overview
  ⊞ 📁 Process Tracing [E2E Process Tracing.xml]
  ⊟ 📁 Services
    ⊟ 📁 GetProductService
      ⊞ 📁 Classes
      ⊟ 📁 Ports
        📦 GetProductPortType
        ⊞ 📦 GetTitlePortType
 	
          Create Element ▸
          Create Diagram ▸
          Create Relation ▸
          Specification   Enter
          Go To ▸
        📐 Select in Inheritance Tree
          Open in New Tree
          Related Elements ▸
          Refactor ▸
          Tools ▸
          Stereotype
          Rename   F2
	Click the new port type in the containment tree with the right mouse button and select **Stereotype** from the context menu. Another menu opens, containing a text field and a list of stereotypes.

Select Stereotype: e2es *4 matches found* ☐ 🔲 E2ESAPRFCModule [Class] ☐ 🔲 E2ESAPTRFCCallbacks [Class] ☐ 🕐 E2EScheduler [Class] ☑ 🔲 **E2ESOAPPortType [Class]** [Apply] [Clear All] [Order]	Port types need the stereotype **E2ESOAPPortType**. In order to locate it in the list of stereotypes you can filter the list by entering the initial letters of the stereotype name, for instance **e2es**. Select the checkbox next to the stereotype **E2ESOAPPortType** and confirm your choice with the **Apply** button.

Containment ⬒ 📌 ✕
📂 ⭐ 🔍   ⚙ ▾
⊟ 🔺 **Data**
  📁 Aliases
  ⊞ 📁 Base Components [E2E Bridge Base.xml]
  ⊞ 📁 Base Types [E2E Bridge Base.xml]
  ⊞ 📁 Component View
  ⊞ 📁 Overview
  ⊞ 📁 Process Tracing [E2E Process Tracing.xml]
  ⊟ 📁 Services
    ⊟ 📁 GetProductService
      ⊞ 📁 Classes
      ⊟ 📁 Ports
        🔲 GetProductPortType
        ⊞ 🔲 GetTitlePortType
	Note, that the icon of class **GetProductPortType** has changed to a port type icon after assigning the stereotype.

Double-click the **GetProductPortType** and navigate to the **Documentation /Hyperlinks** section in the left panel of the dialog.

In the **Documentation** field, enter a description of the port type:

The port type provides an operation to copy a title string and a product record. The data is displayed in the client application.

Click **Close**.

Defining the Port Type Operation

Create an operation for the new port type.

Assign the name **getProduct**.

According the SOAP standard, operations of port types needs to defined as static. In the object-oriented world this means that no instance of the port type class is needed to call the operation. The operation can be called directly without instantiating the port type class (more details are described in the following chapters).
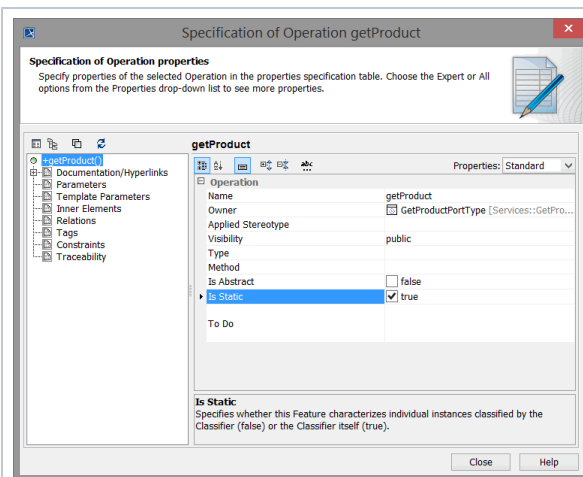
Double-click the new operation **getProduct** to open the operations specification dialog. Set the **Visibility** to **public** and select the checkbox **Is Static** to set this option to **true**.
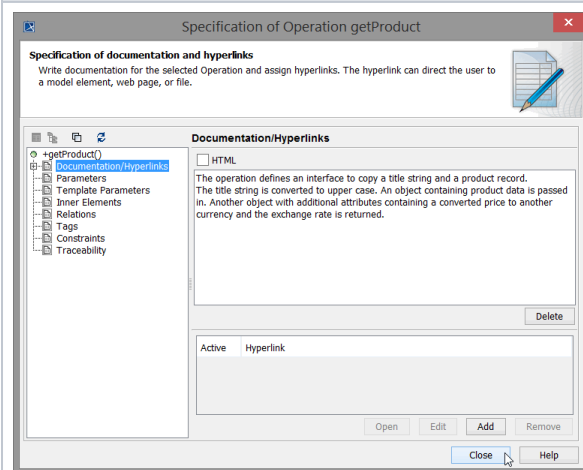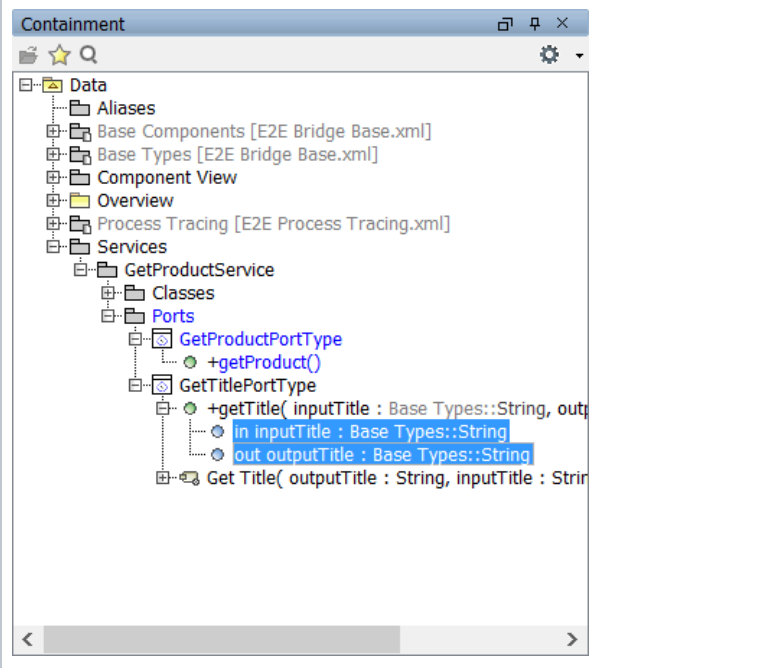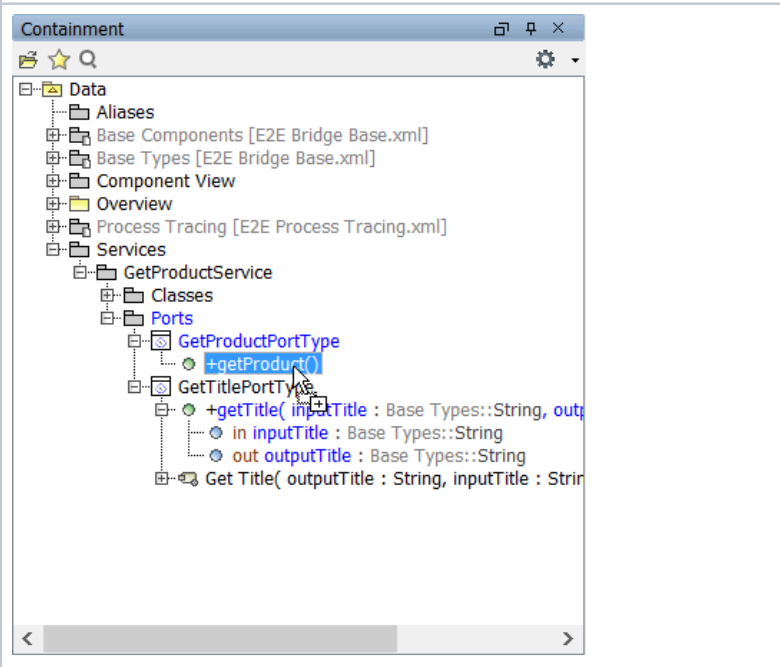

Switch to the **Documentation/ Hyperlinks** section, and enter a description in the **Documentation** field:

The operation defines an interface to copy a title string and a product record. The title string is converted to upper case. An object containing product data is passed in. Another object with additional attributes containing a converted price to another currency and the exchange rate is returned.
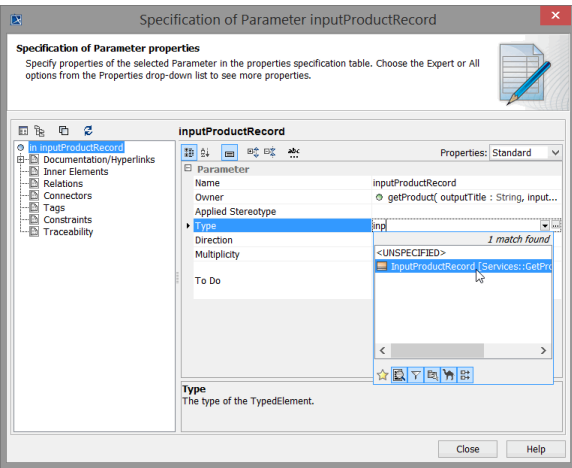
Click **Close**.
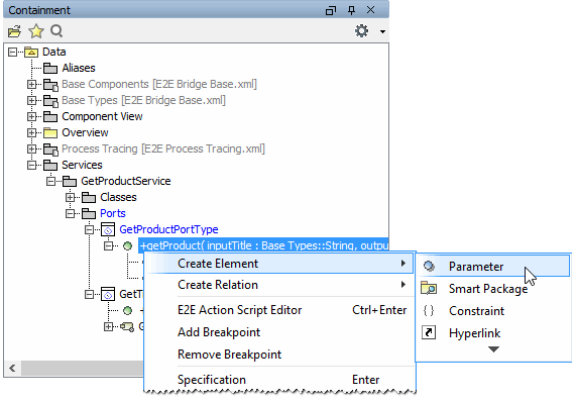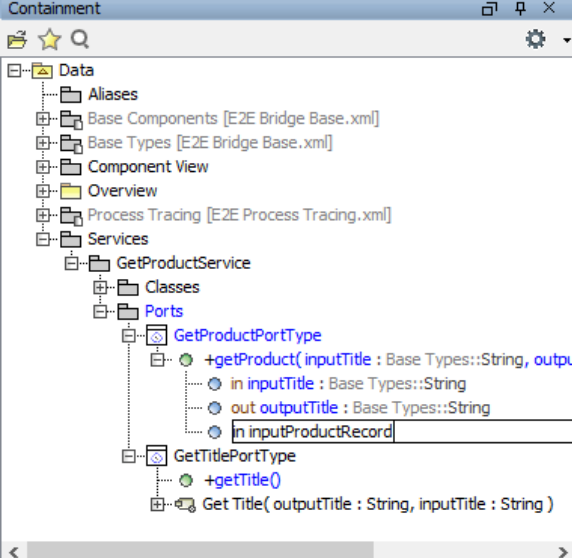
Defining the Operation Parameters

The Web service will be capable of taking a record from the actor and passing this data back to the actor. A data record represents a product. So, the service needs additional operation parameters: one input parameter (**inputProductRecord**) and one output parameter (**outputProductRecord**). As the new port type operation will still convert the title string, too, it also needs the parameters **inputTitle** and **outputTitle**.

For the title parameters, the most efficient way is to copy the parameters in the containment tree from the lesson 1 port type **GetTitleP ortType** to the new port type.

Expand the tree of port type **GetTitleP ortType** and its operation **g etTitle** and select the two parameters **in putTitle** and **o utputTitle**.

Next, keep the **Ctrl** key pressed and drag the two parameters to the operation **getProduct**.

Holding down the **Ctrl** key tells MagicDraw to copy the parameters instead of moving them from one operation to the other. A **+** sign attached to the cursor arrow indicates the copy action.

Release the mouse button when the operation **getProduct** is active. MagicDraw will create an exact copy of the operation parameters.

Using this method, MagicDraw not only copies the parameter names, but also copies all the parameter properties like type and direction.

Create a third operation parameter.

Assign the name **inputProductRecord**.

Double-click the parameter **inputProductRecord** to open the **Parameter** specification dialog.

Click into the **Type** field and start typing **inp** in order to filter the list. Select the previously defined class **InputProductRecord**.
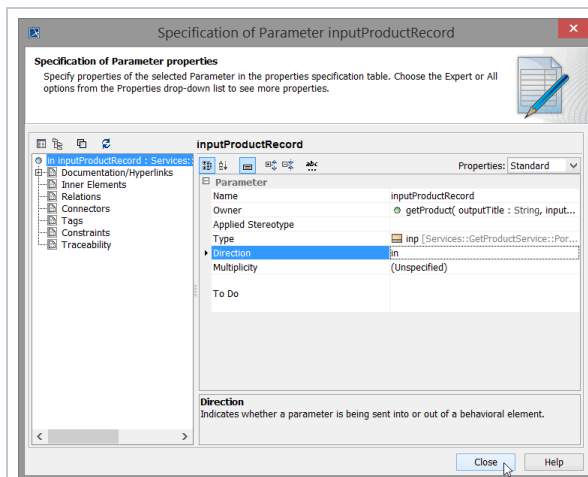
As this parameter is used as input to the operation, the **Direction** has to be set to **in**. This is the default value and has already been set.

Close the dialog.
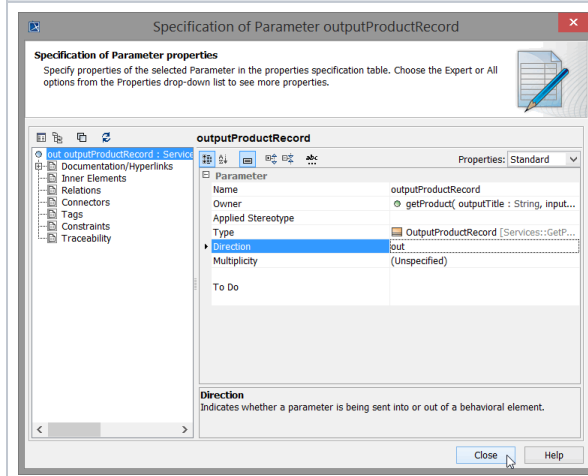

Finally, create another parameter, name it **outputProductRecord**, and choose the type **OutputProductRecord**. Specify the parameter as an **out**put parameter.
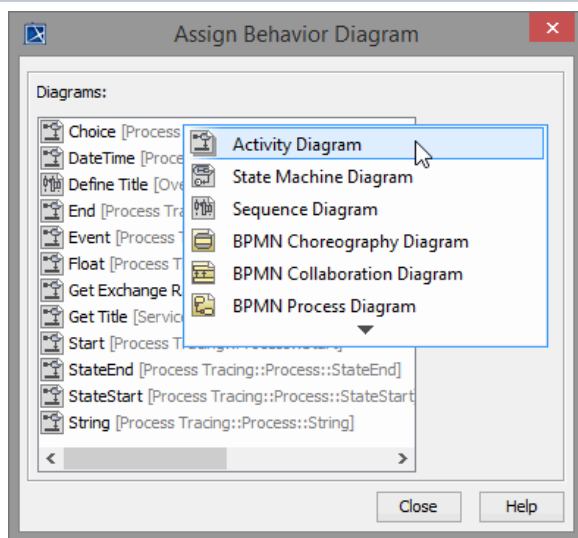
Assigning a New Activity Diagram to the Operation

At this point, the interface of the Web service is nearly complete. Each operation must be assigned to an activity diagram of the UML model. Operations of a port type represent the interfaces of a Web service. Activity diagrams implement the behavior of these operations. Thus, each port type operation has to be assigned to the implementing activity diagram. When the operation is called, the assigned activity diagram will be executed.

The activity diagram has not been created yet. In the next step, you will directly assign a new activity diagram to the operation **getProduct**.
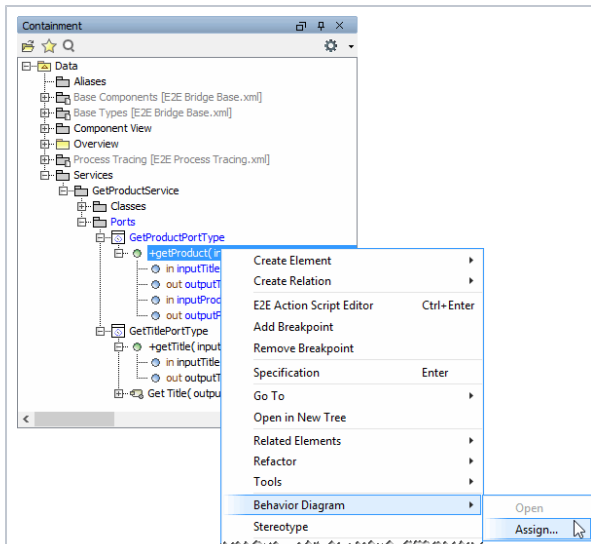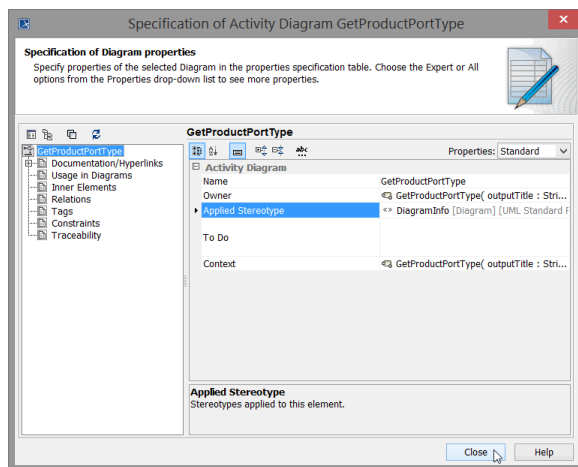
Select the operation **getProduct** in the containment tree with the right mouse button, choose **Behavior Diagram**, and select the menu item **Assign….**
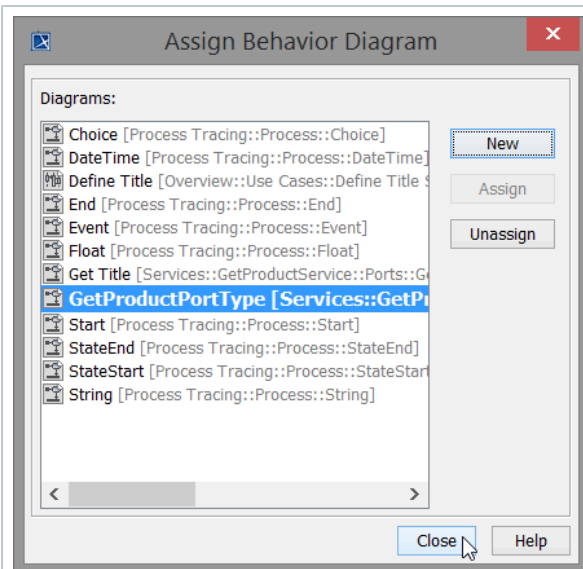

The **Assign Behavior Diagram** dialog displays a list of existing activity diagrams that can be assigned to the operation. However, the operation gets assigned a new activity diagram.

Click **New** and select **Activity Diagram** from the drop down list.
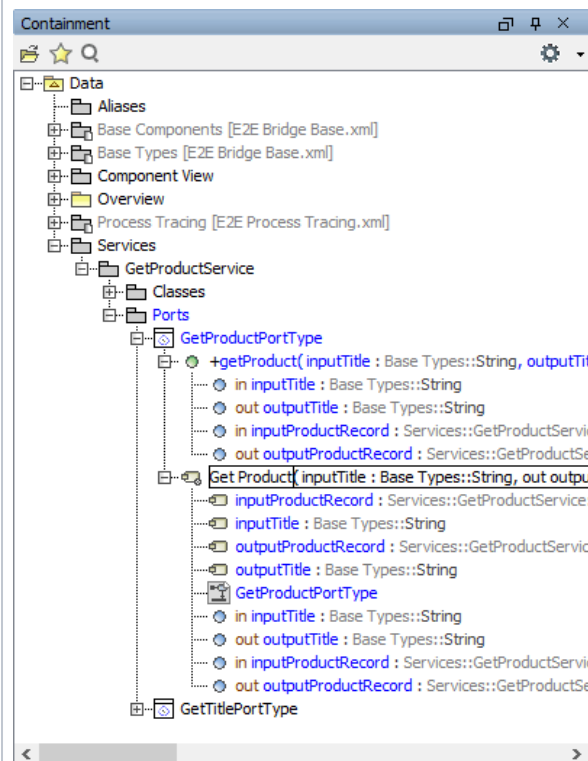

MagicDraw automatically creates an activity diagram and gets the name form the context of the opereation: **GetProductPortType**.

Click **Close**.

The new activity diagram **GetProductPortType** is now listed in the **Assign Behavior Diagram** dialog and is assigned to the operation.

Click **Close**.

Expand the activity **GetProductPortType** in the containment tree. The name of the activity respectively the activity diagram should always correspond to the name of the operation it is specifying.

Rename the activity to **Get Product**. MagicDraw will rename the activity diagram as well.

If you double-click the operation **getProduct** in the containment tree, the assigned activity diagram **Get Product** will be opened in the diagram pane.

The definition of the second SOAP interface of the Web service is completed.

Save the UML model.