How to Draw SoaML Diagrams?

Service Oriented Architect (SOA) allows for describing and understanding how people, organization and system components work together, using services to achieve business objectives. SoaML is an Object Management Group (OMG) standard that provides a domain neutral modeling language to architect and model SOA using Unified Modeling Language (UML).

September 3, 2013

SoaML in VP-UML

VP-UML supports the modeling of SOA with SoaML. In VP-UML, SoaML profile is organized into five SoaML diagram types, called the Service Interface Diagram, Service Participant Diagram, Service Contract Diagram, Services Architecture Diagram and Service Categorization Diagram. Each of them provides a unique view in describing and understanding services and the services architecture. Combined with the use of UML diagrams like sequence diagram, activity diagram, BPMN business process diagram and OMG business motivation model (BMM), you can describe SOA as well as to indicate its technical and business relevance.

What is this Tutorial about?

This tutorial is written to explain what SoaML is, how to use SoaML for SOA and how to draw the various SoaML diagrams in VP-UML. There are mainly 5 parts in this tutorial. Each part explains one of the SoaML diagram type in details, with SoaML tool description, diagram definition and the steps to create the diagram.

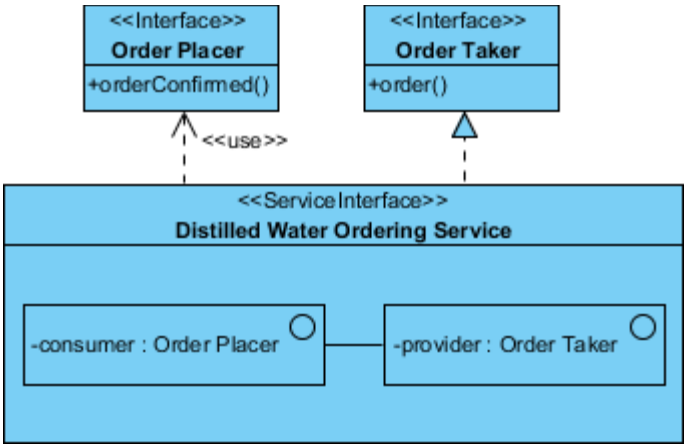
The example that will be used in this tutorial is a basic version of a distilled water supplier. You will draw different SoaML diagrams to explain the different aspects of the distilled water ordering and delivery service.

Preparation

In order perform the steps in this tutorial, make sure you have the professional or enterprise edition of VP-UML downloaded and installed. You may click here to download VP-UML if you do not have it installed.

To avoid disrupting your production environment during the tutorial, please create a new project to perform the steps in this tutorial.

Part I - Drawing Service Interface Diagram



Service Interface Diagram Tools

- Service Interface
- Interface
- Role
- Connector
- Capability
- Expose
- Dependency
- Realization
- Usage
- Message Type
- Milestone

What is Service Interface Diagram?

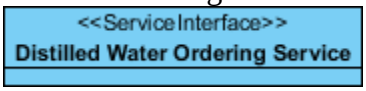
Service Interface Diagram is one of the most important SoaML diagram type. In order to understand what service interface diagram is, you must first know about a key concept of SoaML - service.

As specified in the SoaML specification, service is "value delivered to another through a well-defined interface". In SoaML, a service can be specified using three approaches: simple interface, service interface and service contract. Simple interfaces define one-way services that do not require protocol. Such services can be used with anonymous callers and the participant know nothing about the caller. Service interface allows for bi-directional services. Such services involve the communication between provider and consumer of services in completing services. Service contract defines how participants work together to exchange value and we will talk about it when introducing the service contract diagram.

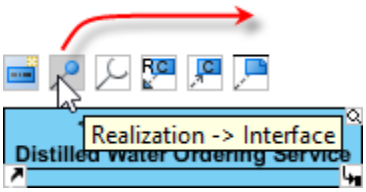
Service interface diagram allows for the modeling of service specification. You can model simple interfaces and service interfaces in service interface diagram. Now, follow the steps below to draw a service interface diagram.

How to Draw a Service Interface Diagram?

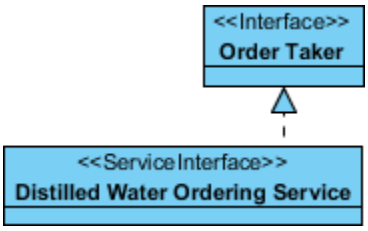
1. In a new project, create a service interface diagram by selecting **SoaML > Service Interface Diagram** from the toolbar.
2. We are going to create a service interface for the distilled water ordering service. Select **Service Interface** from the diagram toolbar, under the category **Service Interface**.
3. Click on the diagram to create a service interface. Name it *Distilled Water Ordering Service*.



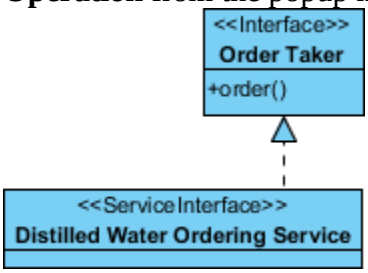
4. We are going to define the interface for the provider of the distilled water ordering service. Move the mouse pointer over the service interface Distilled Water Ordering Service. Drag out the resource **Realization -> Interface**.



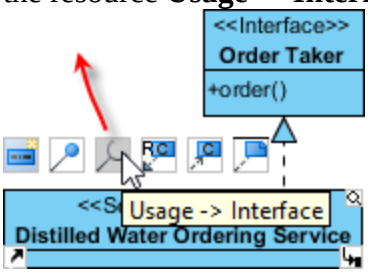
5. Release the mouse button at the top right of the service interface. Name the created interface *Order Taker*.



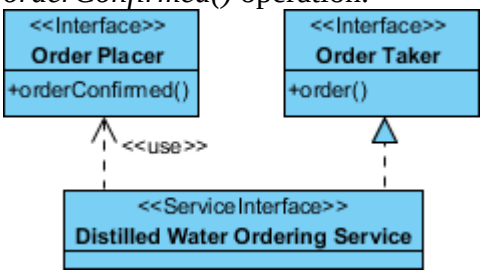
6. Provider interface contains operations that may be invoked during the course of service. It is a must for service provider to support the operation defined. Order taker is responsible for processing customers' order. Add an operation *order()* in *Order Taker* by right clicking on *Order Taker* and selecting **Add** > **Operation** from the popup menu.



7. We are going to define the interface for the consumer of the distilled water ordering service. Drag out the resource **Usage -> Interface** from the service interface *Distilled Water Ordering Service*.

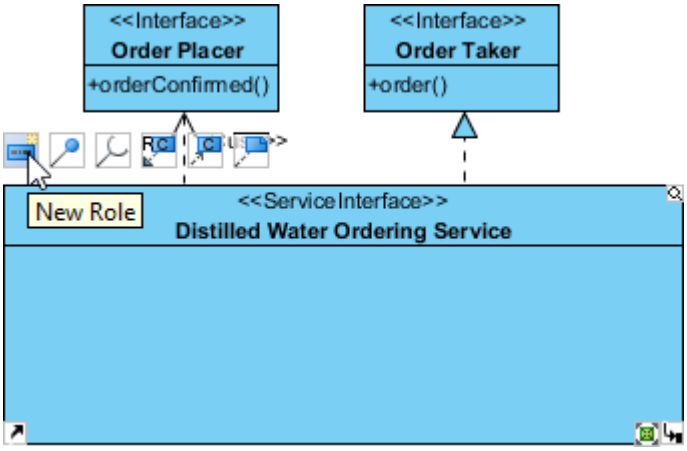


8. Name the interface *Order Placer*.
9. Add an operation *orderConfirmed()* in it. This indicates that consumer of service must support the *orderConfirmed()* operation.

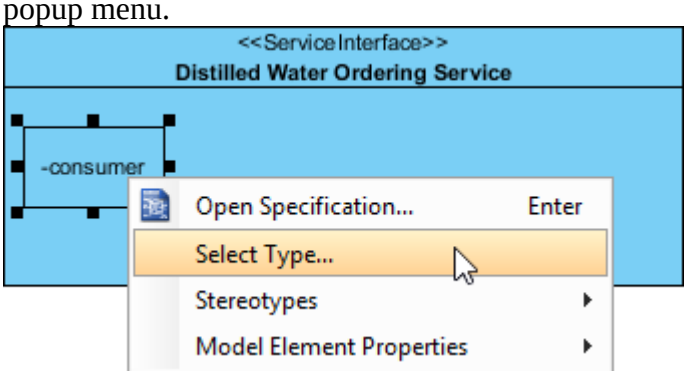


10. Let's define the roles of provider and consumer in the distilled water ordering service. The roles have to be defined inside the body of service interface. So let's resize the service interface *Distilled Water Delivery Service* to make it bigger.

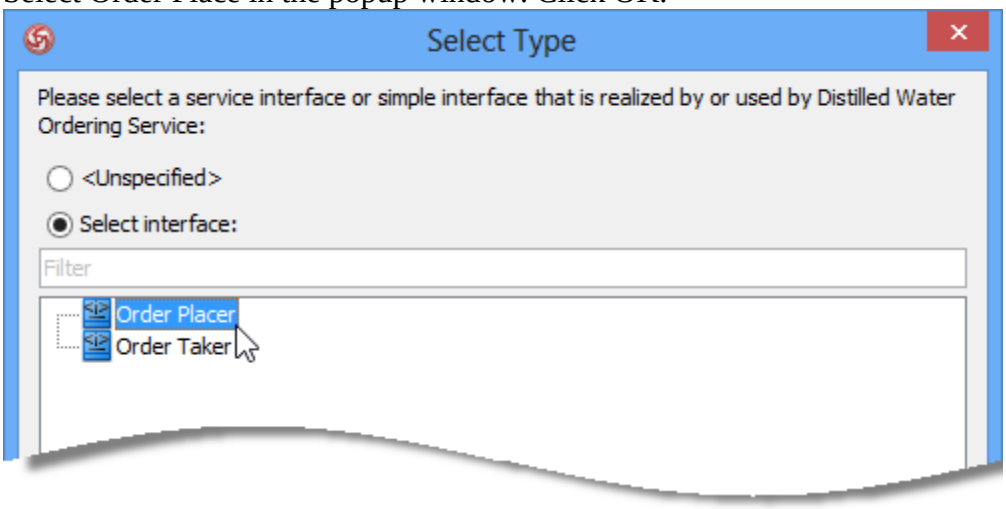
11. Use the **New Role** resource icon to create a role in *Distilled Water Delivery Service*.



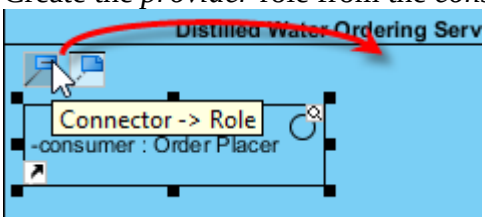
12. Name the role *consumer*.
13. The type of consumer role is *Order Placer*. Right click on the role and select **Select Type...** from the popup menu.



14. Select Order Place in the popup window. Click OK.

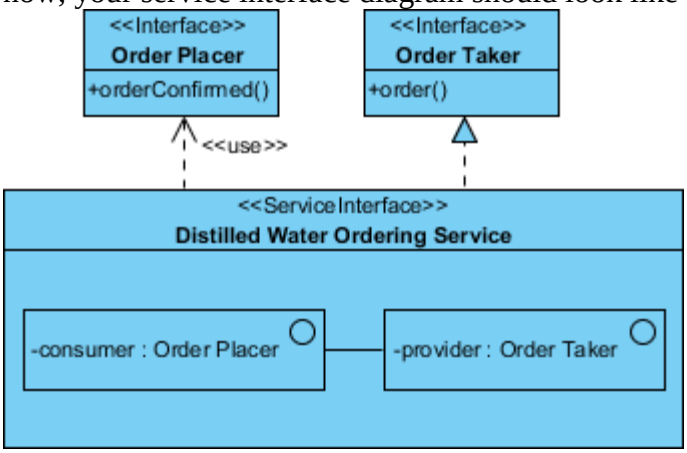


15. Create the *provider* role from the *consumer* role, through the use of the resource-centric interface.

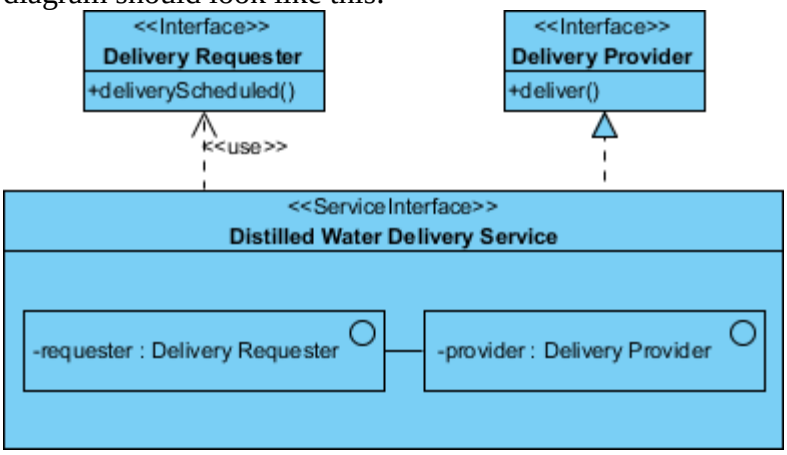


16. Name the role *provider*.

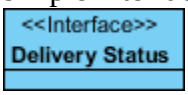
17. Similar to how you set type for the consumer role, set *Order Taker* to be the type of provider role. Up to now, your service interface diagram should look like this:



18. You've completed defining the distilled water ordering service. Now, apply the skills you've learned in the previous steps to define the distilled water delivery service in a new service interface diagram. Your diagram should look like this:



19. Before we end this section, let's also try to create a simple interface. As mentioned before, simple interface defines a one-way service that require no protocol. Create a new service interface diagram first.
20. Let's create a simple interface for querying the order delivery status. Click on the diagram to create a simple interface and name it *Delivery Status*.



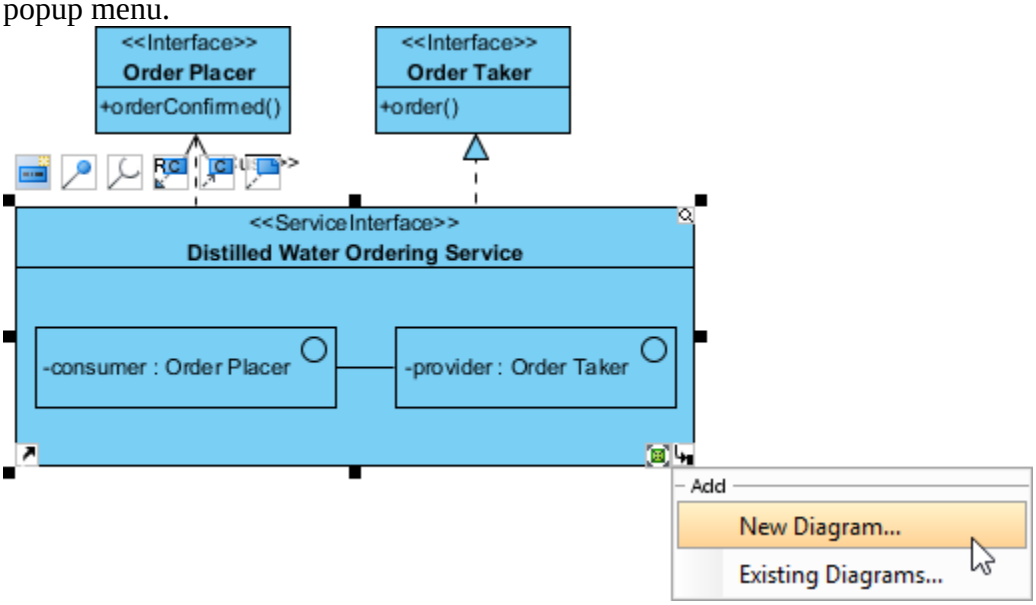
21. Add an operation *getDeliveryStatus(id:String):String* in it.

Part II - Specifying Choreography Using UML Sequence Diagram

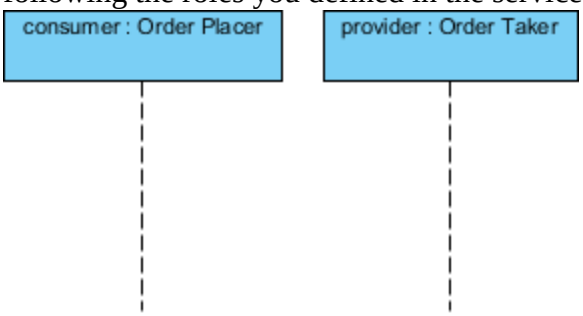
Service choreography defines the interaction between the provider and consumer in completing a service. A UML sequence diagram can be used for specifying service choreography. In VP-UML, you can add sub-sequence-diagram to service interface for such purpose. Now, try to specify the service choreography for the service interface *Distilled Water Ordering Service* defined in the previous section.

1. Open the first service interface diagram where the service interface *Distilled Water Ordering Service* was defined.
2. Click on the service interface *Distilled Water Ordering Service*.

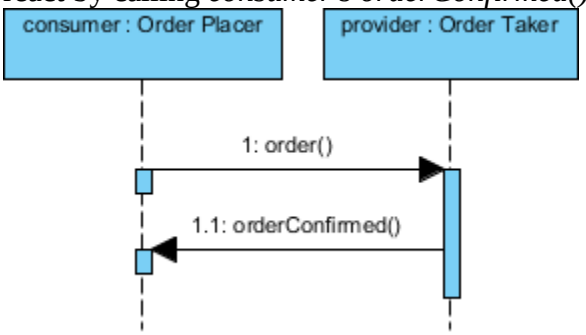
3. Click on the tiny resource icon at the bottom right of the shape and select **New Diagram...** from the popup menu.



4. The **New Diagram** window is opened, with **Sequence Diagram** automatically selected.
5. Click **OK** to confirm diagram creation.
6. A sequence diagram is created, with lifelines *consumer* and *provider* in it. The lifelines are created by following the roles you defined in the service interface.

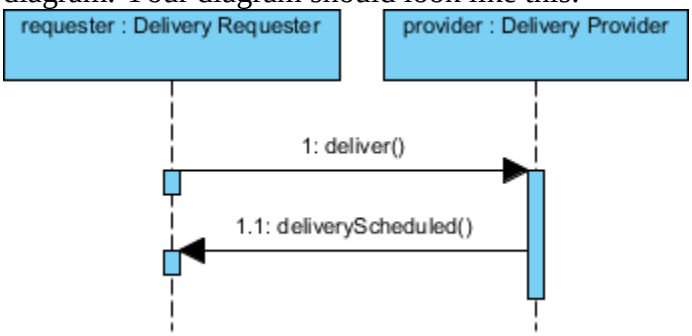


7. Specify how the consumer interact with the provider of service by drawing sequence messages between the two lifelines. The *consumer* begin by invoking *provider's order()* method. The *provider* will then react by calling *consumer's orderConfirmed()* method.



In practice you can specify optional call with the use of opt fragment.

8. Now, try to specify the choreography of the distilled water delivery service, using UML sequence diagram. Your diagram should look like this:

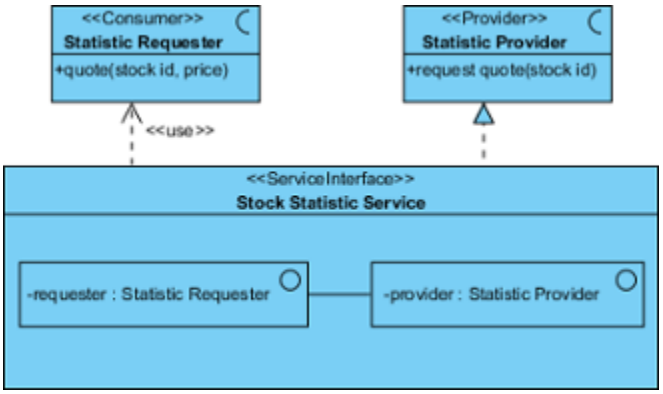


Creating service interface diagram

SoaML supports both a contract and an interface-based approach to SOA. They differ in the way services are specified.

The interface-based approach involves the use of simple interfaces and service interface. Simple interface focuses mainly on one-way service delivery that requires no protocol between parties. Service interface allows for bi-directional services. Provider and consumer work together to complete a service.

Service interface diagram is a type of SoaML diagram specialized for the definition and specification of both simple interface and service interface.



A sample service interface diagram

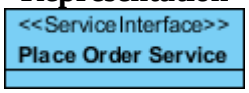
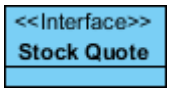
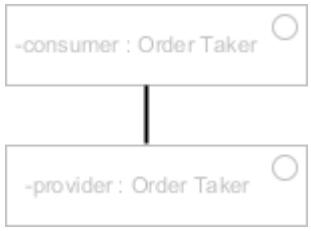
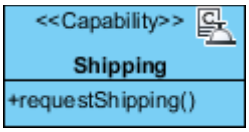
Creating service interface diagram

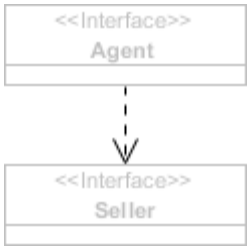
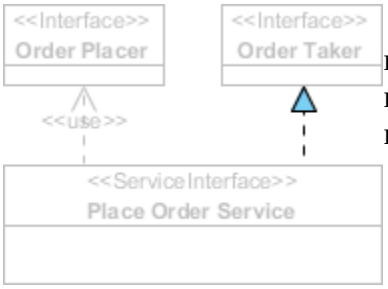
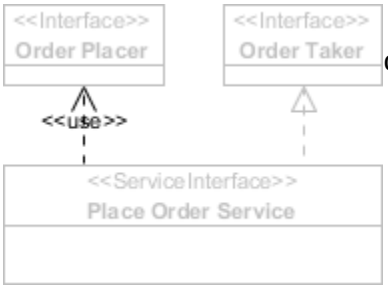
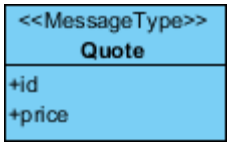
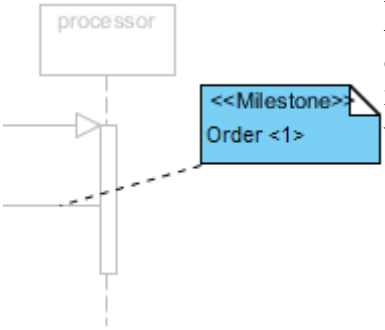
To create a SoaML service interface diagram, take any of the steps below:

- Click on **SoaML** on toolbar and select **Service Interface Diagram**
- Right click on **Service Interface Diagram** in **Diagram Navigator** and select **New Service Interface Diagram** from the popup menu.
- Select **File > New Diagram > SoaML > Service Interface Diagram** from the main menu.

Notations

The description of notations is either extracted or derived from the OMG SoaML Specification v1.0.1.

Name	Representation	Description
Service Interface		A ServiceInterface defines the interface and responsibilities of a participant to provide or consume a service. It is used as the type of a Service or Request Port. A ServiceInterface is the means for specifying how a participant is to interact to provide or consume a Service. A ServiceInterface may include specific protocols, commands, and information exchange by which actions are initiated and the result of the real world effects are made available as specified through the functionality portion of a service. A ServiceInterface may address the concepts associated with ownership, ownership domains, actions communicated between legal peers, trust, business transactions, authority, delegation, etc.
Interface		Simple interfaces define one-way services that do not require a protocol. Such services may be defined with only a single UML interface and then provided on a "Service" port and consumed on a "Request" port.
Role		A ServiceInterface is a UML Class and defines specific roles each participant plays in the service interaction. These roles have a name and an interface type. The interface of the provider (which must be the type of one of the parts in the class) is realized (provided) by the ServiceInterface class. The interface of the consumer (if any) must be used by the class.
Connector		Connect roles in a service interface.
Capability		A Capability models the ability to act and produce an outcome that achieves a result that may provide a service specified by a ServiceContract or ServiceInterface irrespective of the Participant that might provide that service. A ServiceContract, alone, has no dependencies or expectation of how the capability is realized – thereby separating the concerns of 'what" vs. "how." The Capability may specify dependencies or internal process to detail how that capability is provided including dependencies on other Capabilities. Capabilities are shown in context using a service dependencies diagram.
Expose		The Expose dependency provides the ability to indicate what Capabilities that are required by or are provided by a participant should be exposed through a Service Interface.

Dependency		The Provider may also have a uses dependency on the consumer interface, representing the fact that the provider may call the consumer as part of a bi-directional interaction. These are also known as "callbacks" in many technologies.
Realization		A ServiceInterface specifies the receptions and operations it receives through InterfaceRealizations. A ServiceInterface can realize any number of Interfaces. Some platform specific models may restrict the number of realized interfaces to at most one.
Usage		A ServiceInterface specifies its required needs through Usage dependences to Interfaces.
Message Type		A MessageType is a kind of value object that represents information exchanged between participant requests and services. This information consists of data passed into, and/or returned from, the invocation of an operation or event signal defined in a service interface. A MessageType is in the domain or service-specific content and does not include header or other implementation or protocol-specific information.
Milestone		A Milestone depicts progress by defining a signal that is sent to an abstract observer. The signal contains an integer value that intuitively represents the amount of progress that has been achieved when passing a point attached to this Milestone.

A list of supported notations in service interface diagram