

# Adding parameters to entities



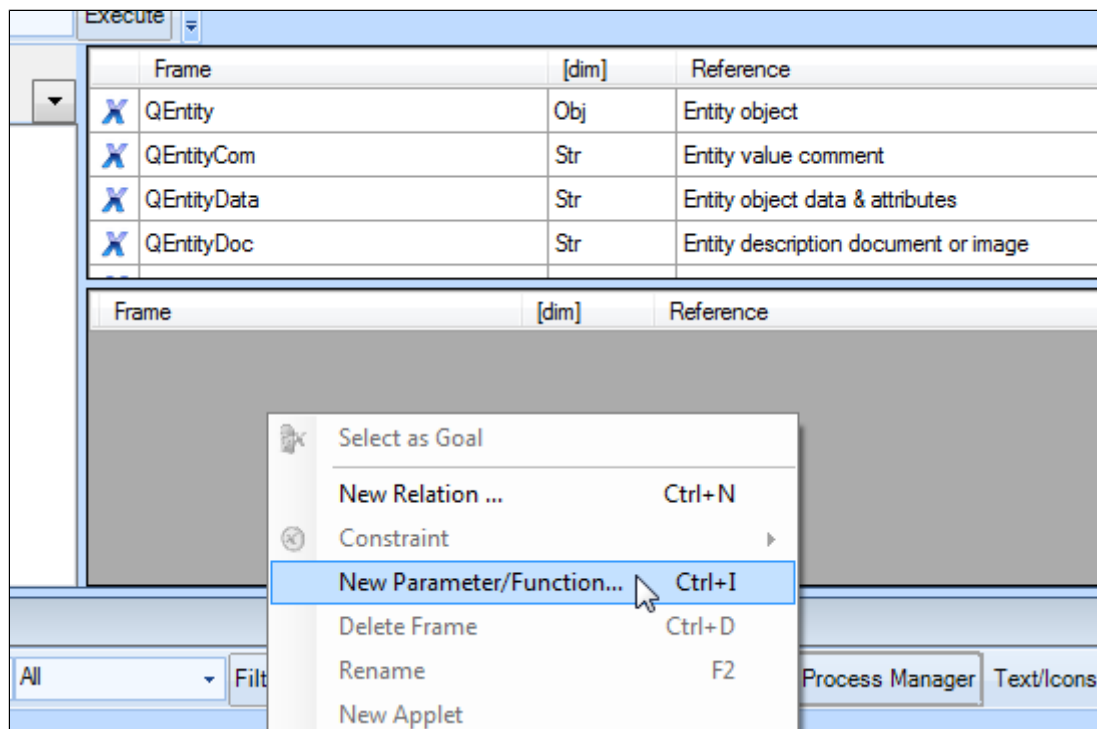
## 1 Adding parameters to Main Dimensions

The entity `Main Dimensions` is the first one that you will add contents to. These contents consist of Quaestor parameters.

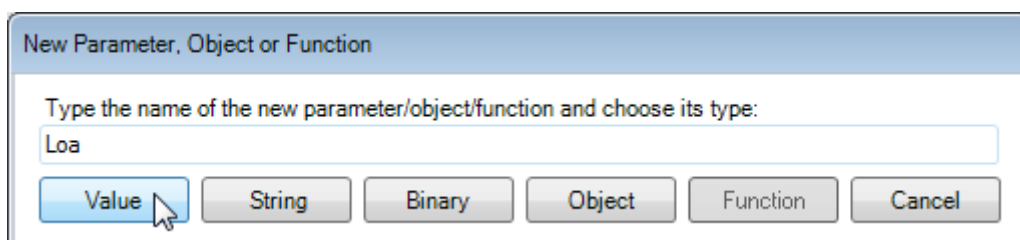
During the intended calculation, the user will be asked to provide input values in entity `Main Dimensions` for the following parameters:

Parameter name	Dimension	Reference (i.e. comment)
Loa	[m]	Length over all
Lpp	[m]	Length between perpendiculars
Boa	[m]	Width over all
Dm	[m]	Moulded depth of ship


- Right-click in the right field of the **Knowledge Browser** and select *New Parameter/Function...*, or press **Ctrl+I**.



A small window opens, where you can enter the name and type of the parameter:




- Enter the name `Loa` and click on *Value*.

 A valid parameter name should not contain special characters and spaces.

The created parameter `L0a` is placed in the **Knowledge Browser** with a red cross in front of it.

If you have experience with creating 'classical' (i.e. taxonomy-free) Quaestor knowledge bases (see the [Tutorials on Quaestor basics](#)) you will know that a parameter should always have a unique name and a dimension and the system should "know" how to determine the value of the parameter.

- Select `L0a` in the **Knowledge Browser**.
- In the **Properties** window, enter `m` in the *Dimension* field, `Length over all` in the *Reference* field, and set *Determined by* to *VR: User only*.


 A special remark has to be made concerning the *Determined by* field for use with taxonomies. When you are sure that parameters should be requested as input, change the *Determined by* field to *VR*. When you add an entity-relation or connect a relation to a parameter (see next section), leave the *Determined by* field to *USR* or *USL*. Quaestor will then make sure the added or connected relation is used and the red cross for the parameter will disappear the moment the relation is added or connected.

- In the same way, add the *VR* parameters `Lpp`, `Boa` and `Dm` in the **Knowledge Browser**. All have `m` as dimension.

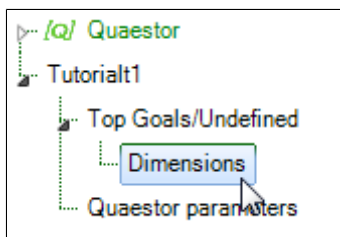


## 2 Creating classes

You can create tree nodes in the **Knowledge Browser**, in order to group parameters and relations.

 Classes are just a way to order and group your knowledge and have no functional meaning during the use of knowledge.

- *Right-click* on the `Top Goals/Undefined` node in the tree of the **Knowledge Browser** and select *New Class*. Name it `Dimensions`.



When you have created classes, you can simply drag and drop the parameters and/or relations to the desired class.

- Drag the parameters that you have just created to the `Dimensions` class.

If you define a parameter while a certain class has the focus, the parameter will belong to that class.



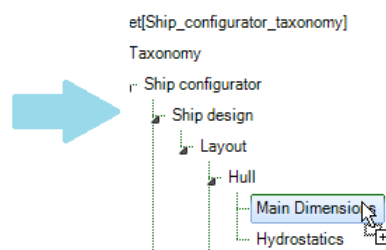
### 3 Including parameters in an entity

Now that you have created parameters in the **Knowledge Browser**, the next step is to include these parameters in the entity **Main Dimensions**.

Two ways are available to include parameters in an entity:

1. Include parameters by a drag/drop action between **Knowledge Browser** and **Workbase**.
  2. Use the option *Include Parameters from knowledge base* while creating a new entity.
- Using the first method, drag parameter **Lpp** to the entity **Main Dimensions** (under **Layout > Hull**):

	Frame	[dim]	Reference	Statu
<input checked="" type="checkbox"/>	Boa	[m]	Width over all	
<input checked="" type="checkbox"/>	Dm	[m]	Moulded depth of ship	
<input checked="" type="checkbox"/>	Loa	[m]	Length over all	
<input checked="" type="checkbox"/>	Lpp	[m]	Length between perpendiculars	



- Do the same for **Loa**, **Boa** and **Dm**.

The second method to include parameters in an entity, is by means of the select option *Include Parameters from knowledge base* while in the entity editor. After closing the entity editor, Quaestor will present a list of parameters that are defined in the knowledge base. All selected parameters will be included in the newly created entity. This is especially convenient when a lot of parameters are already available in the knowledge base the moment you create a particular entity.

[Back to content](#) | [<<Previous](#) | [Next >>](#)