Knowledge Advisor

Overview

This book is intended for the user who needs to become quickly familiar with Knowledge Advisor.

This overview provides the following information:

- Knowledge Advisor in a Nutshell
- Before Reading this Guide
- Getting the Most out of this Guide
- Accessing sample documents
- Conventions Used in this Guide

Knowledge Advisor in a Nutshell

CATIA - KNOWLEDGE ADVISOR is a *CATIA* product which allows users to embed knowledge within design and leverage it to assist in engineering decisions, in order to reduce errors or automate design, for maximum productivity.

Users can embed knowledge in design such as formulas, rules and checks and leverage it when required at any time. Knowledge is then taken into account and acts according to its definition. Its meaning is also accessible: For example a check intent can highlight the parameters involved in a verification, it is easy and immediate to understand in what way a standard has been violated.

In short, Knowledge Advisor enables users to:

- Capture corporate engineering knowledge as embedded specifications allowing complete consistency.
- Easily define and share know-how among all users.
- Automate product definition.
- Ensure compliance with corporate standard.
- Increase productivity.
- Increase Knowledge management for sharing and understanding intents.
- Build Knowledge components management for customization and reuse.
- Allow early attention to final design specifications preventing costly redesigns.
- Guide and assist users through their design tasks.

Before Reading this Guide

Before reading this guide, you should be familiar with basic Version 5 concepts such as document windows, standard and view toolbars. Therefore, we recommend that you read the *Infrastructure User's Guide* that describes generic capabilities common to all Version 5 products. It also describes the general layout of V5 and the interoperability between workbenches.

Getting the Most out of this Guide

To get the most of this guide, we suggest that you start performing the step-by-step Getting Started tutorial.

Once you have finished, you should move on to the User Tasks section.

The Workbench Description section, which describes the Knowledge Advisor workbench, and the Customizing section, which explains how to set up the options, will also certainly prove useful.

Accessing sample documents

To perform the scenarios, you will be using sample documents contained in either the online/kwrug/samples folder.

For more information about this, please refer to Accessing Sample Documents in the *Infrastructure User's Guide*.

Conventions Used in this Guide

To learn more about the conventions used in this guide, refer to the *Conventions* section.

Conventions

Certain conventions are used in CATIA, ENOVIA & DELMIA documentation to help you recognize and understand important concepts and specifications.

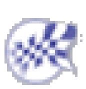
Graphic Conventions

The three categories of graphic conventions used are as follows:

- Graphic conventions structuring the tasks
- Graphic conventions indicating the configuration required
- Graphic conventions used in the table of contents

Graphic Conventions Structuring the Tasks

Graphic conventions structuring the tasks are denoted as follows:

This icon...	Identifies...
	estimated time to accomplish a task
	a target of a task
	the prerequisites
	the start of the scenario
	a tip
	a warning
	information
	basic concepts
	methodology
	reference information
	information regarding settings, customization, etc.
	the end of a task

	functionalities that are new or enhanced with this release
	allows you to switch back to the full-window viewing mode

Graphic Conventions Indicating the Configuration Required

Graphic conventions indicating the configuration required are denoted as follows:

This icon...	Indicates functions that are...
P1	specific to the P1 configuration
P2	specific to the P2 configuration
P3	specific to the P3 configuration

Graphic Conventions Used in the Table of Contents

Graphic conventions used in the table of contents are denoted as follows:

This icon...	Gives access to...
	Site Map
	Split View mode
	What's New?
	Overview
	Getting Started
	Basic Tasks
	User Tasks or the Advanced Tasks
	Workbench Description
	Customizing
	Reference
	Methodology
	Glossary

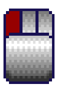
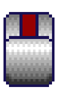
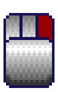
Text Conventions

The following text conventions are used:

- The titles of CATIA, ENOVIA and DELMIA documents *appear in this manner* throughout the text.
- **File** -> **New** identifies the commands to be used.
- Enhancements are identified by a blue-colored background on the text.

How to Use the Mouse

The use of the mouse differs according to the type of action you need to perform.

Use this mouse button...	**Whenever you read...**
	• Select (menus, commands, geometry in graphics area, ...) • Click (icons, dialog box buttons, tabs, selection of a location in the document window, ...) • Double-click • Shift-click • Ctrl-click • Check (check boxes) • Drag • Drag and drop (icons onto objects, objects onto objects)
	• Drag • Move
	• Right-click (to select contextual menu)

What's New?

New Functionality

Handling Errors

It is possible to test a feature in error when creating rules.

Creating a Loop in a Reaction

It is possible to use the For and While constructs in Knowledge Advisor actions and reactions.

3D PLM Integration

Saving a Product Structure Containing a Rule in ENOVIA VPM V5

The user can now store relations at the Product level in Enovia. This example provides the user with a rule example.

Getting Started

Before getting into the details for using *CATIA* - Knowledge Advisor Version 5, this section provides a step-by-step scenario demonstrating how to use Knowledge Advisor key functionalities. You should be familiar with the basic commands common to all workbenches. These are described in the *Infrastructure User's Guide*.

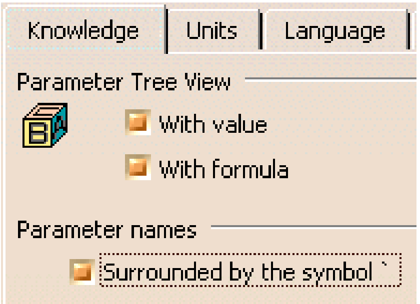
When working in a Japanese environment, remember to check the **Surrounded by the Symbol'** option (**Tools->Options->General->Parameters and Measure->Knowledge** tab).

Using Parameters
Using Formulas
Using Rules
Using Checks

Using Parameters

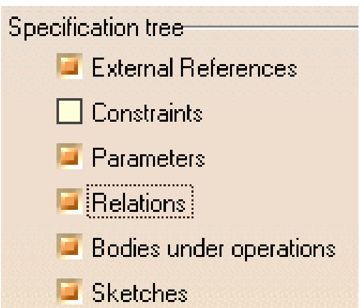
This task explains how to use parameters. For a fuller outline of the parameters-related tasks, see the Knowledgeware Infrastructure - Tips and Techniques - Summary dedicated to the knowledgeware infrastructure capabilities.

Check the settings below:



- From the Tools menu, select **Options->General->Parameters and Measure**.
- In the **Knowledge** tab, check the **With Value** and **With Formula** check boxes, and click **OK**.

When working in a Japanese environment, check the **Surrounded by the symbol'** check box under Parameter names.

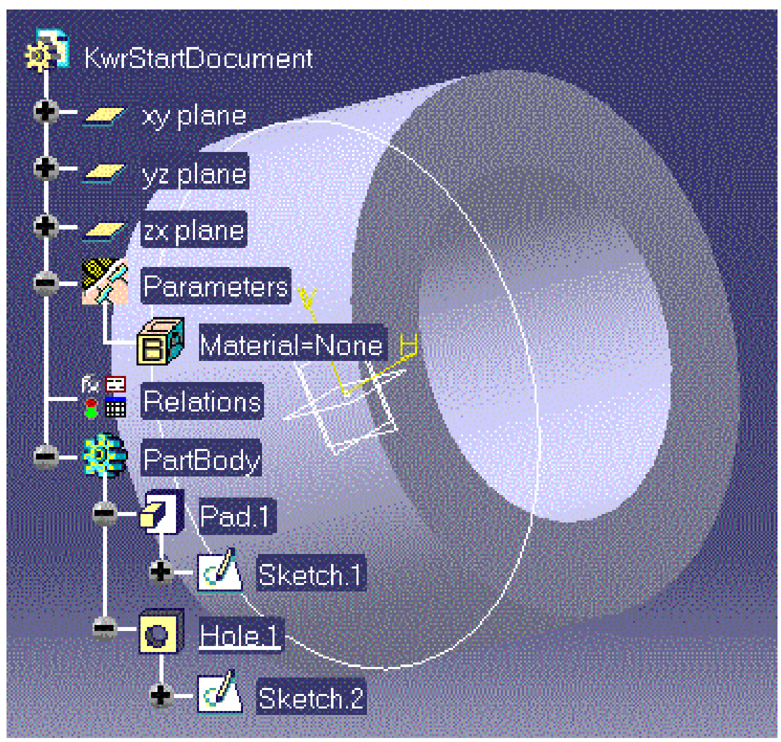


- From the **Tools** menu, select **Tools->Options...->Infrastructure->Part Infrastructure.**
- Check at least the **Relations** and **Parameters** boxes in the **Display** tab, and click **OK**.

It is recommended to check all the options located below the Specification tree settings.

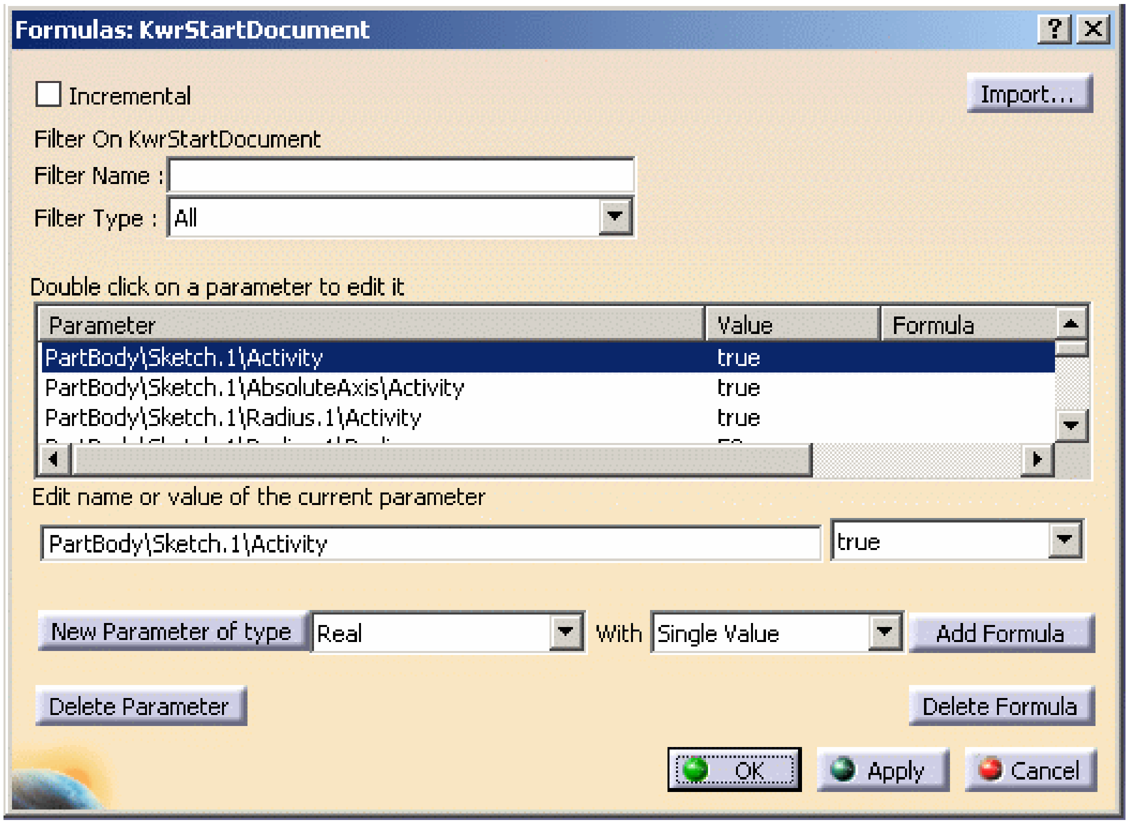
1. Open the KwrStartDocument.CATPart document.

If you expand the Parameters node in the specification tree, the Material parameter is the only one displayed.
At this stage of the scenario, don't pay any attention to this default parameter.



The Relations node can't be expanded as there is no default relation in a *CATIA* document.

2. Click the  icon. The Formulas dialog box is displayed.



3. In the **New Parameter of type** scrolling list, select the **Length** type, then click the **New Parameter of type** button.

4. In the **Edit name or value of the current parameter** field, replace the Length.1 string with PadLength, and click **Apply**.

   A new parameter is added to the document parameter list both in the Formulas dialog box and in the specification tree.

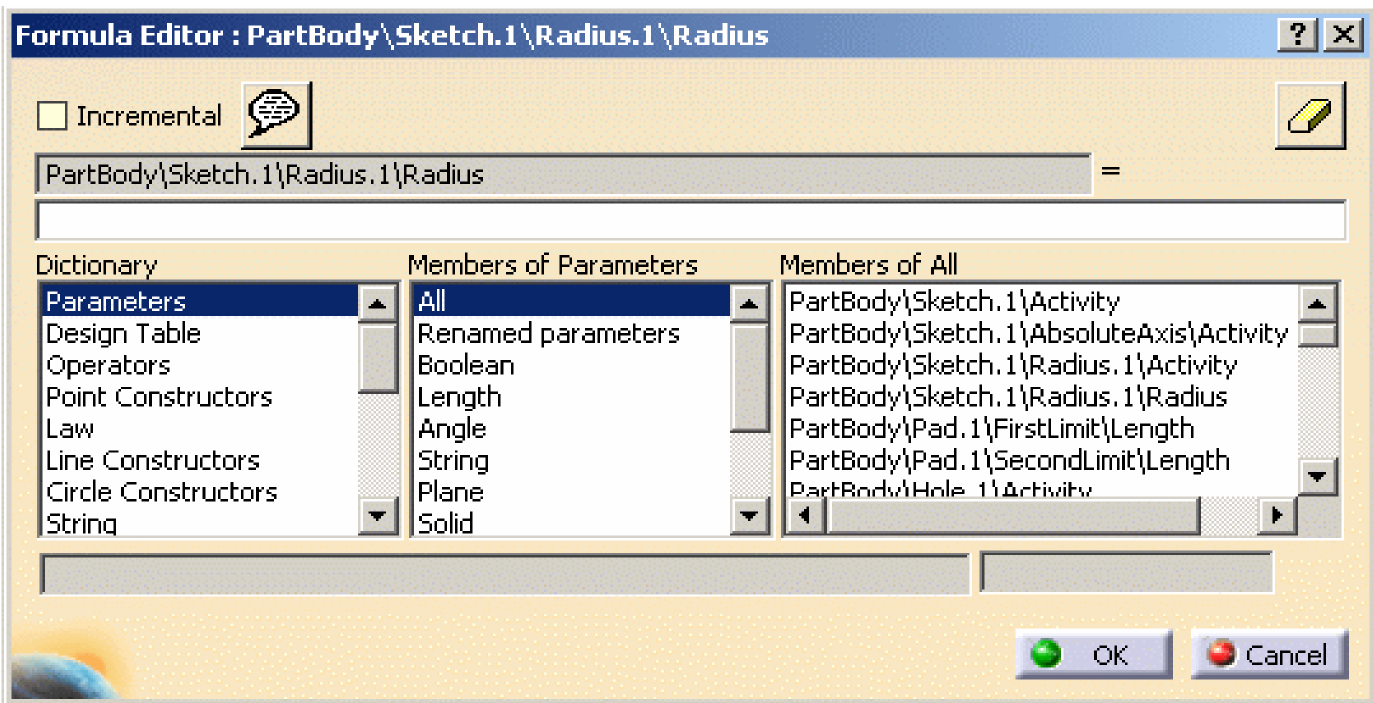
   You have just created a *user parameter*.

5. Click **OK** in the Formulas dialog box to terminate the dialog. Keep your document open and proceed to the next task.

Using Formulas

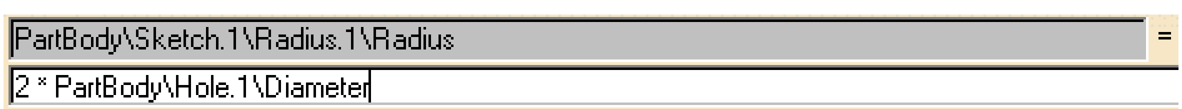
This task explains how a parameter can be constrained by a formula.
See the Knowledgeware Infrastructure - Tips and Techniques - Summary dedicated to the infrastructure knowledgeware capabilities for more information on formulas.

1. Click the f(x) icon. The Formulas dialog box is displayed.
2. In the parameter list, select the PartBody\Sketch.1\Radius.1\Radius item, then click **Add Formula**. The Formula editor displays.



The icon located on the right is simply a rubber you can use to erase the formula.

3. Enter the 2 * PartBody\Hole.1\Diameter relation.



4. Click **OK** in the **Formula Editor** once you have typed your relation. The Formula.1 relation is added to the specification tree.

   In the parameter list of the dialog box, a formula is now associated with the sketch radius.
5. In the parameter list, select the PadLength item, click **Add Formula** to create the formula

below:

PadLength = PartBody\Pad.1\FirstLimit\Length + PartBody\Pad.1\SecondLimit\Length

In the parameter list, the Formula.2 relation is now associated with the PadLength user parameter. In the specification tree, PadLength is also displayed with the value resulting from Formula.2.

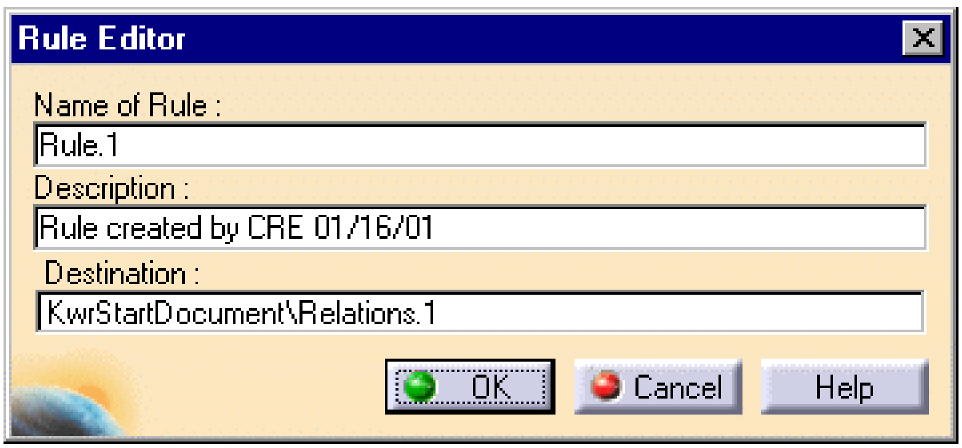
6. Click **OK** twice in the Formulas dialog box to terminate this task. Keep your document open and proceed to the next task. This is now what you should see in the specification tree under "Relations":

Using Rules

This task introduces the Knowledge Advisor rules.
Unlike the parameter and formula capabilities which are available to all *CATIA* users, the rule and check capabilities require the Knowledge Advisor product.

For more information about Rules, see Working with the Rule Feature. To know more about the Rule Editor, see Using the Rule Editor.

1. Select the KwrStartDocument item in the specification tree
2. Access the Knowledge Advisor workbench from the **Start->Knowledgeware** menu.
3. Click the rule icon. The following dialog box is displayed:



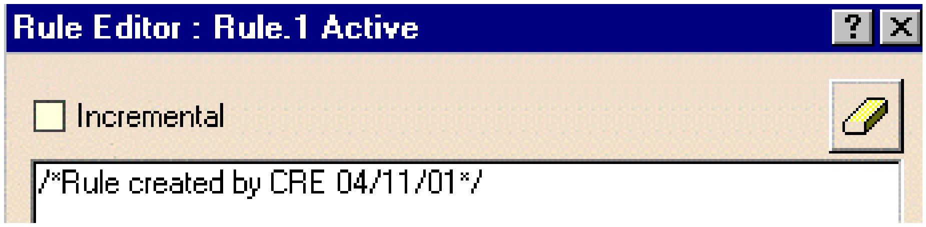
The dialog box fields display default values that can be modified:

a - The rule name: Rule.i. The first rule created in a document is Rule.1 by default. This name is the one displayed in the specification tree unless you modify the default name at creation.

b - The user and the date of creation.

c - The destination, i.e. the feature you are going to add the rule to. By default, in this scenario, the destination is the Relations feature (the Relations node in the specification tree). But a rule could be added to another feature, then only apply to this feature.

4. Replace the Rule.1 string with Cylinder_Rule, if need be modify the comments but don't modify the destination. Click **OK**. The Rule Editor is displayed (see below).



5. Type the code below into the edition box or copy/paste it from your browser to the edition box.

```
PartBody\Hole.1\Activity = true
if PadLength <= 50mm and PadLength > 20mm
{
PartBody\Hole.1\Diameter = 20mm
Message("PadLength is: # | Internal Diameter is: #",
PadLength,PartBody\Hole.1\Diameter)
}
else if PadLength > 50mm and PadLength < 100mm
{
PartBody\Hole.1\Diameter = 50mm
Message("PadLength is: # | Internal Diameter is: #",
PadLength,PartBody\Hole.1\Diameter)
}
else if PadLength >= 100mm
{
PartBody\Hole.1\Diameter = 80mm
Message("PadLength is: # | Internal Diameter is: #",
PadLength,PartBody\Hole.1\Diameter)
}
else
{
PartBody\Hole.1\Activity = false
Message("PadLength is: # | Internal Diameter is: #",
PadLength,PartBody\Hole.1\Diameter)
}
```

Users working in a Japanese environment should use the script below:

```
`PartBody\Hole.1\Activity` = true
if `PadLength` <= 50mm and `PadLength` > 20mm
{
`PartBody\Hole.1\Diameter` = 20mm
Message("PadLength is: # | Internal Diameter is: #",
`PadLength`,`PartBody\Hole.1\Diameter`)
}
else if `PadLength` > 50mm and `PadLength` < 100mm
{
`PartBody\Hole.1\Diameter` = 50mm
Message("PadLength is: # | Internal Diameter is: #",
`PadLength`,`PartBody\Hole.1\Diameter`)
}
else if `PadLength` >= 100mm
{
`PartBody\Hole.1\Diameter` = 80mm
Message("PadLength is: # | Internal Diameter is: #",
```

```
`PadLength`,`PartBody\Hole.1\Diameter`)
}
else
{
`PartBody\Hole.1\Activity` = false
Message("PadLength is: # | Internal Diameter is: #",
`PadLength`,`PartBody\Hole.1\Diameter`)
}
```

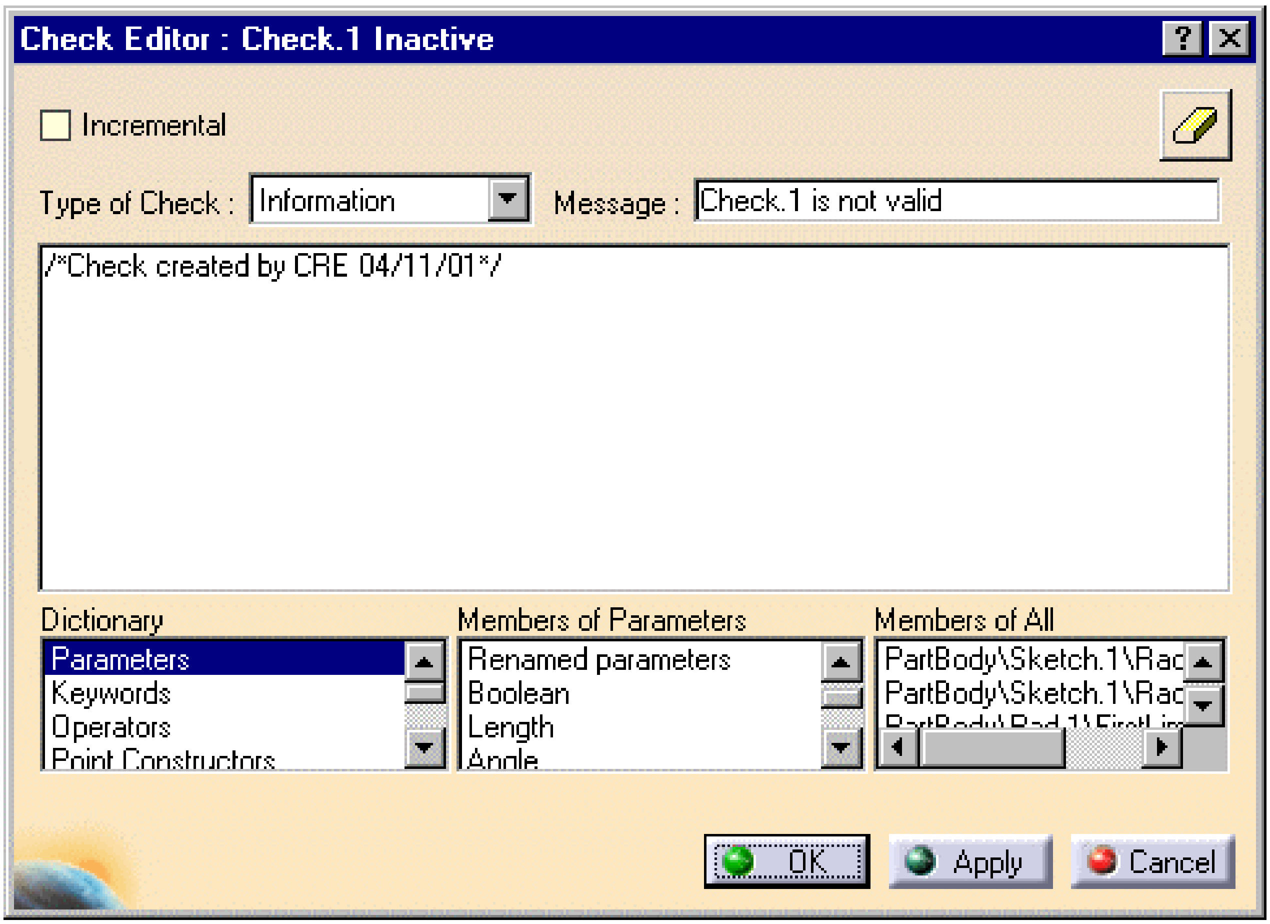
6. Click **Apply**. An information window displays the PadLength and Pad internal diameter values. Click OK in the Information window. The Cylinder_Rule relation is added to the specification tree.

7. Click **OK** to terminate this part of the dialog. Keep your document open and proceed to the next task.

Using Checks

This task explains how to create a check and what happens when you add a check to a document.
The Knowledge Advisor product is required for this task.
See the Rule and Check Tasks for more information on check-related tasks.

1. Click the icon. The first "Check Editor" dialog box is displayed.
2. Replace the Check.1 default name with Cylinder_Check, then click **OK**. The **Check Editor box** is displayed. It is similar to the Rule Editor. The **Incremental** box must be unckecked.



3. Select the Information item in the **Type of Check** list.
4. Enter a string in the message field (for example: Pad too short). This message is to be displayed whenever the statement specified by the check is not fulfilled.
5. Enter the following statement into the edition box: PadLength > 20mm
6. Click OK to confirm the check creation. The Cylinder_Check relation is added to the

specification tree. A green icon in the specification tree means that the check is fulfilled. No message is displayed.

7. Change the Pad limits so that PadLength <= 20mm. The Cylinder_Rule relation is re-applied. An information window displays the new PadLength and Pad internal diameter values. Then, you are warned by another window ("Pad too short") that the check is no longer valid. The check icon in the specification tree turns to red.

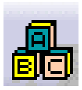
User Tasks

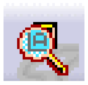
Refer to the Quick Reference of Tasks for a comprehensive list of interactions to be carried out on rules and checks. See also the Useful Tips, the Limitations, and the CATIA Knowledgeware Infrastructure sections.

This section shows you how to manage the Knowledge Advisor relations.

Working with Parameters

 Select the Formula icon to create parameters.

 Select the Add Set of Parameters icon to create sets of parameters. These sets of parameters are all grouped below the Parameters node.

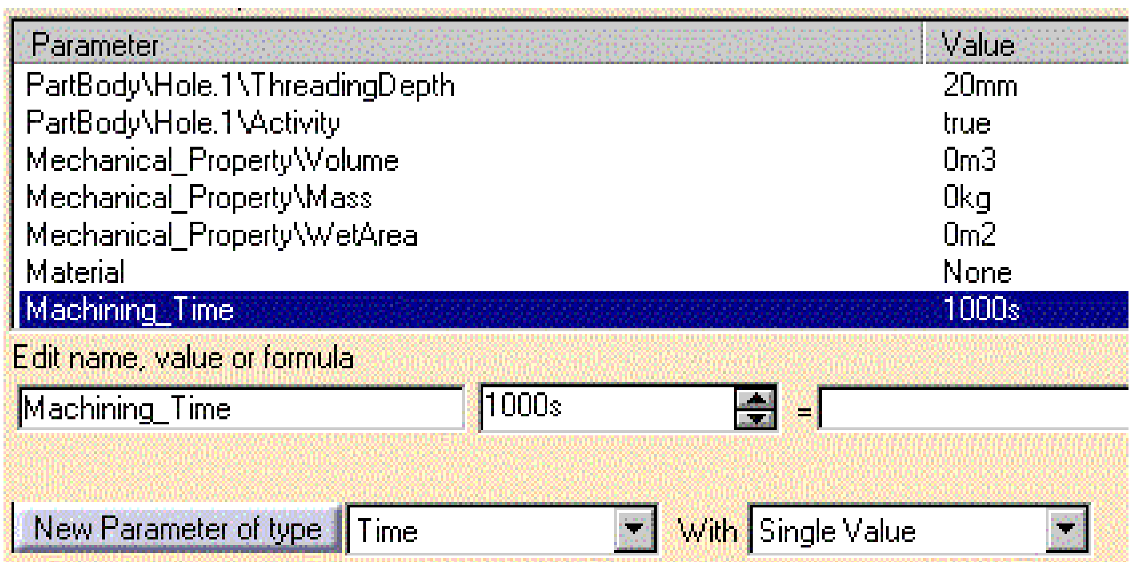
 Select the Parameters Explorer icon to add new parameters to a feature.

 Select the Add parameters on geometry icon to add new parameters to a face, a vertex, or an edge.

Creating a Parameter

This task explains how to create a Time type parameter and assign a value to it.

1. Open the KwrStartDocument.CATPart document.
2. Click the icon. The *f(x)* dialog box is displayed.
3. Select the **Time** item with **Single Value** in the **New Parameter of type** list, then click **New Parameter of type**. The new parameter appears in the Edit name or value of the current parameter field.
4. Replace the Time.1 name with Machining_Time and assign the 1000s value to this parameter. Then click **Apply**. The Machining_Time parameter is added to the specification tree. The dialog box is modified as follows:



5. Click **OK** when done to close the dialog box.

- You can add properties to a .CATPart or a .CATProduct document by using the Properties command from the contextual menu. You just have to click the Define other properties... button in the Product tab then click New parameter of type. The dialog is similar to the *f(x)* dialog. See the *Product Structure User's Guide* for more information. The properties you define that way are also displayed in the parameter list of the *f(x)* dialog box.

- You can specify that a parameter is constant by using the Properties command from the contextual menu. This command also enables you to hide a parameter.

Introducing Parameters

When you create a part like the hollow cylinder of our "Getting Started" example, you often start by creating a sketch, then you create a pad by extruding the initial sketch, then you add other features to the created pad. The final document is made up of features which define the intrinsic properties of the document. Removing one of these features results in a modification of the document. These features are called ***parameters***. Parameters play a prominent role in knowledgeware applications. They are features that can be constrained by relations and they can also be used as the arguments of a relation.

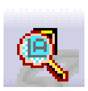
In addition to these parameters, *CATIA* allows you to create ***user parameters***. These user parameters are extra pieces of information added to a document.

User parameters are very handy in knowledgeware applications:

- They can be used to add specific information to a document
- They can be defined or constrained by relations
- They can be used as the arguments of a relation.

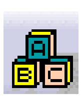
Parameters are created clicking one of the following icons:

The parameters are created using the Formulas Editor. To know more about this editor, see Getting Familiar With the ***f(x)*** Dialog Box.



The parameters are created using the Parameters Explorer Editor. To know more about this editor, see Getting Familiar with the Parameters Explorer.

The created parameters only apply to edges, faces and vertex. The editor is similar to the Parameters Explorer editor.

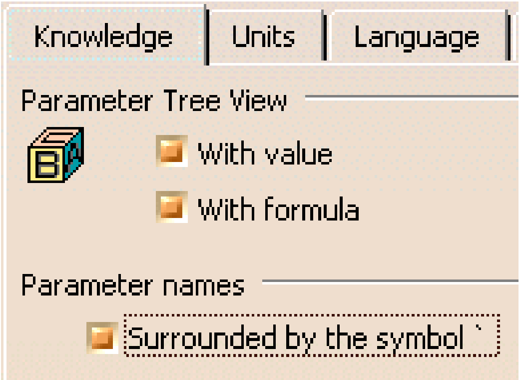


The Set of Parameters enables the user to gather user parameters below a set.

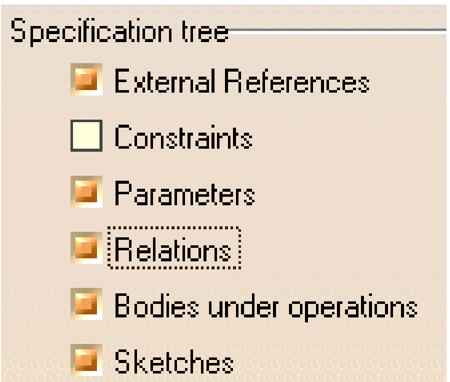
A given relation may take as its arguments both types of parameters (intrinsic and user).

For the parameters to display in the specification tree, check the settings below:

- From the Tools menu, select Options->General->Parameters and Measure.
- In the Knowledge tab, check the With Value and With Formula check boxes, and click OK if you want the parameters to display their values and associated formulas (if any.)



When working in a Japanese environment, check the Surrounded by the symbol' check box under Parameter names.



- From the Tools menu, select Tools->Options...->Infrastructure->Part Infrastructure
- Check at least the Relations and Parameters boxes in the Display tab, and click OK.

Copy/Pasting Parameters

The **Tools->Options->General->Parameters and Measure** check boxes allow you to:

- Paste a parameter without the formula which defines it.
  For example:
  Holeplus= 15 = Diameter + 10 will be pasted as Real.i = 15
  (if the With Value box is checked)

- Paste a parameter as well as the formula which defines it, but only if the parameters referred to in the formula are also selected in the copy.
  For example:
  Holeplus= 15 = Diameter + 10 will be pasted as Real.i = 15 if the Diameter parameter does not belong to the items selected for the copy
  but HolePlus will be pasted as Real.i = 15 = Real.j + 10 if Diameter is selected in the copy (use multi-selection).

- Paste a parameter as well as the formula.
  Holeplus= 15 = Diameter + 10 will be pasted as Real.i = Diameter + 10

When copying parameters sets containing hidden parameters, these parameters are automatically pasted when pasting the parameters sets and appear as hidden parameters.

Specifying the Material Parameter

Whatever your document, the Material parameter is always displayed in the specification tree. The Material parameter is created only after a material is applied to a Part or a Product. The Mechanical_Property features are calculated from the Material value. Specify a material to set the values of the Mechanical_Property features.

1. Open the KwrStartDocument.CATPart document.

   The Material parameter is displayed by default in the specification tree. Its value is set to None.

2. Double-click the Material feature in the specification tree to edit the parameter. The dialog box below is displayed.

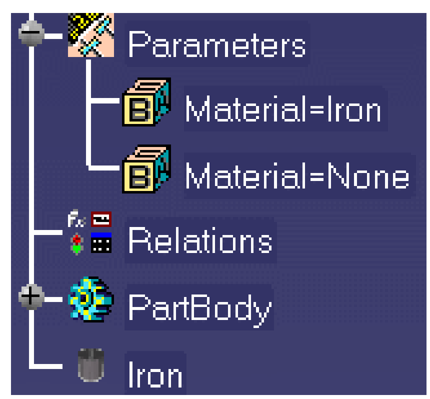
   

3. Click **OK** and select the root feature in the specification tree.

4. Click the  icon in the standard toolbar to display the available material library. Select the **Metal->Iron** material.

5. Click **Apply Material** and **OK**.

   This is what you should see now in the specification tree. The Iron feature is added to the specification tree and a new material is added under the Parameters node.

   Remember: To display parameter values, check **Tools->Options->General->Knowledge->Parameters and Measure->With value**.

6. Keep your document open and proceed to the next task.

Valuating the Mechanical Property Parameters

Once the Material value has been specified, the Mechanical_Property parameters are automatically updated when the Properties option is selected in the contextual menu.

1. Select the root item in the specification tree and open the **Properties** dialog box from the contextual menu.
2. Select the **Mass** tab. The document mechanical properties have been updated from the value assigned to the Material parameter.
3. Click **OK** to go back to your document.

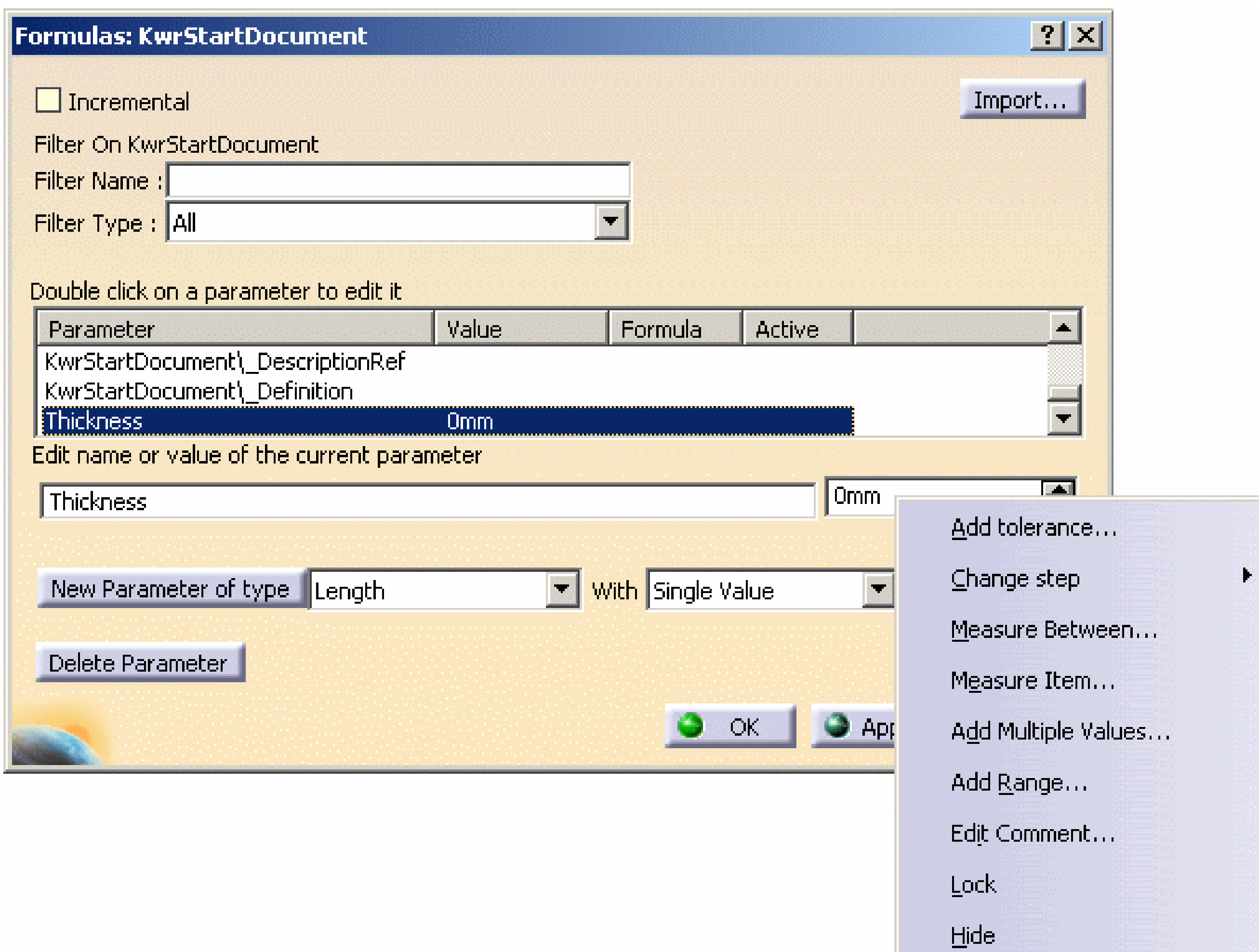
Specifying a Parameter Value as a Measure

This scenario shows how to assign a value to a parameter deducing it from a graphic selection. In this scenario, the user deduces the value assigned to the Thickness parameter by selecting 2 circular edges.

A common way to assign a value to a parameter is to use the Edit name or value of the current parameter field of the Formulas dialog box. But there is another way to proceed. The value you assign to a parameter can be deduced from a graphic selection.

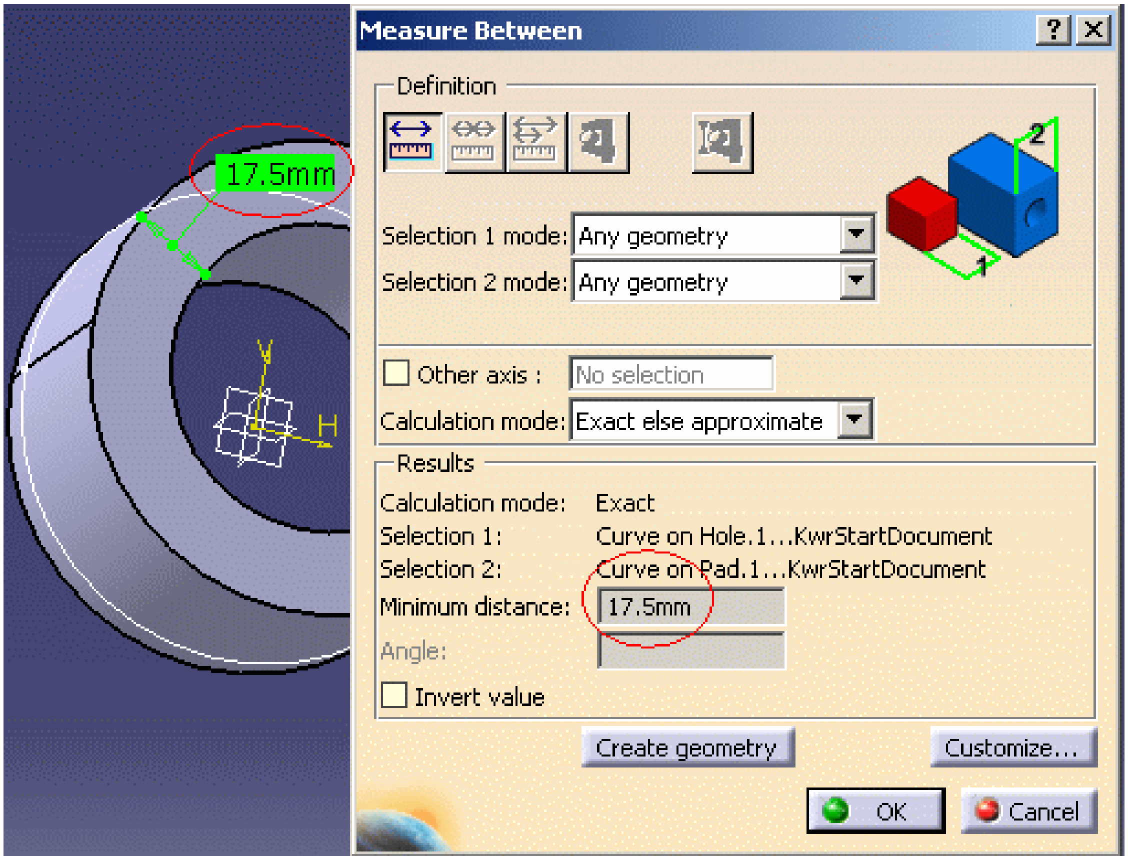
1. In **Tools->Options->General->Parameters and Measure**, check the **Load extended language libraries** box of the Language tab.
2. Open the KwrStartDocument.CATPart document.
3. Click the f(x) icon. The *f(x)* dialog box is displayed.
4. Select the **Length** item with **Single Value** in the **New Parameter of type** list, then click **New Parameter of type**.

   The new parameter appears in Edit name or value of the current parameter.
5. Replace the Length.1 name with Thickness, then right-click in the value field of **Edit name or value of the current parameter**.



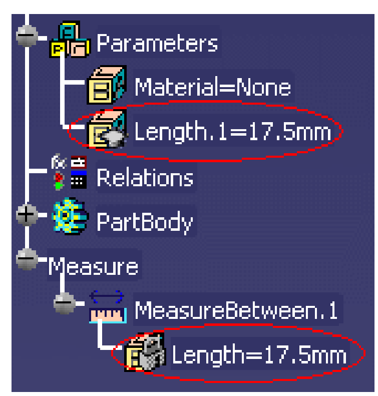
6. Select the **Measure Between...** command from the contextual menu. The Measure Between dialog box is displayed. Select Edge only as **Selection 1** mode and Edge only as **Selection 2** mode.

7. In the document geometry area, select successively one of the inner circular edge of the part, then the outer circular edge located on the same face. The 17.5 mm value is displayed in the Measure Between dialog box.



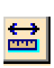
8. Click **OK** when done in the Measure Between dialog box. The 17.5 mm value is displays in the Formulas dialog box.

9. Click **OK** to close the Formulas dialog box.



The parameter displays below the Measure node in the specification tree and below the Parameters node.

To edit this parameter, proceed as follows:

- Double-click it in the specification tree. The Edit Parameters dialog box displays.
- Click the  icon located next to the value field. The Measure between dialog box displays.
- Edit the parameter and click **OK** when done.

Importing Parameters

This scenario shows how to import parameters from an excel or a .txt file into a CATPart document.

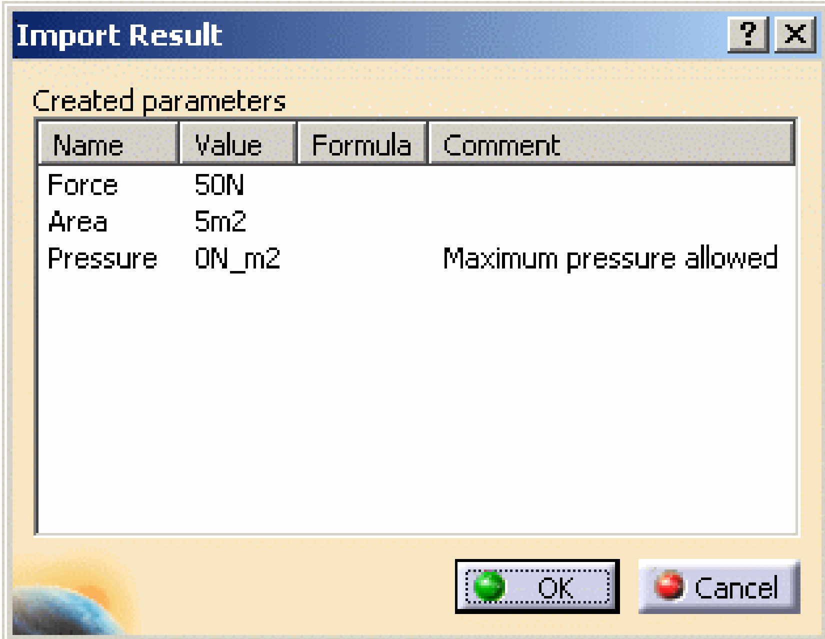
- Parameters and parameter values can be imported from a text file or from an Excel file (Windows) into documents (CATPart, CATProduct, Drawings...).
- If imported parameters already exist in the document, the import process automatically updates the document.

Please find below the formatting rules the external file should comply with:

- **Column 1**
  Parameter names
- **Column 2**
  Parameter values. *Multiple values are allowed*. Values should then be separated by a ";". The imported value is the one delimited by the "<" and ">" tags. Use the Tab key to skip from one column to the other in a tabulated text file.
- **Column 3**
  Formula. If no formula is specified, the third column should be left empty. In a tabulated text file, just press the Tab key twice from column 2 to leave column 3 empty.
- **Column 4**
  Optional comment.

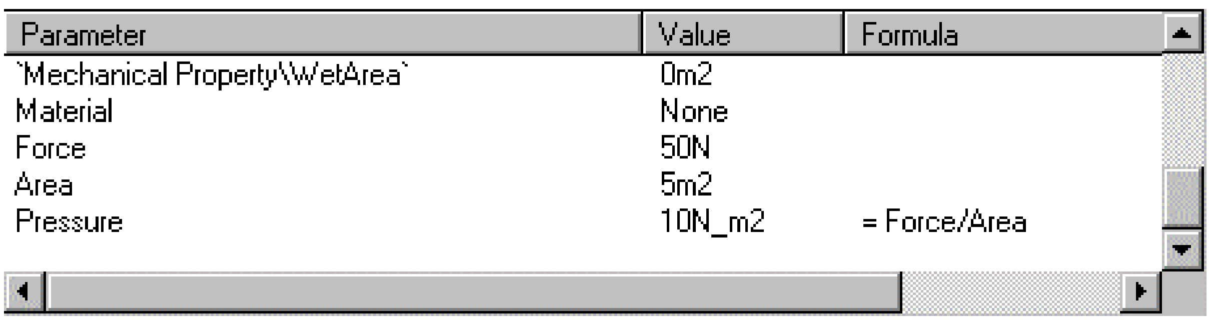
1. Open the KwrStartDocument.CATPart document.
2. Click the f(x) icon. The *f(x)* dialog box is displayed.
3. Click Import.... A file selection dialog box is displayed.
4. Select the ExCompanyFile0.xls file (Windows only) or the TxCompanyFile0.txt file, then click **Open**.

   The list of parameters to be imported into the KwrStartDocument.CATPart document is displayed.



5. Click **OK** to import the parameters from the input file into the KwrStartDocument.CATPart document.

The imported parameters are now displayed in the parameter list of the *f(x)* dialog box and in the specification tree.



6. Click **OK** to terminate the dialog.

Creating Points, Lines... as Parameters

The scenario below explains how to determine the position of the inertia axis of a pad. To do so, start from a pad, then:

1. Create a line by using either method ('datum' or f(x))
2. Use the inertiaAxis line constructor to specify that this line is to be the inertia axis of the pad.
3. Retrieve the coordinate of the point located at the intersection of the inertia axis and the pad extrusion plane.

To create elements such as Points, Lines, Curves, Surfaces, Planes or Circles and use them in knowledgeware relations, you can:

- Create these elements as 'Isolate' elements in the Generative Shape Design workbench. 'Isolate' elements also called *Datum* are elements that have no link to the other entities that were used to create them. For information on 'Datum' type elements, see the *Generative Shape Design User's Guide*.
- Create these elements by using the *f(x)* capabilities and select the right type of element in the New parameter of type list.

1. Access the Part Design workbench, create any sketch in the yz plane, then extrude this sketch to create a pad. If need be, refer to the *Part Design User's Guide*.
2. Create a line intended to be used as an inertia axis afterwards.
3. To do so, click the Formulas icon , select the Line item in New Parameter of type, then click New Parameter of type.
4. Click the Formulas icon. In the parameter list, select the line you have just created (Geometrical Set.1\Line.1).
5. Click Add Formula and add the formula below in the editor:

   Geometrical Set.1\Line.1 = inertiaAxis(3,PartBody)

   The inertiaAxis function is accessible through the Line constructors. The axis number 3 is the one which is in the extrusion direction (normal to yz). Click OK in the Formulas dialog box. The inertia axis is displayed in the geometry area.

6. Back to . Create three length type parameters: X, Y and Z.
7. Retrieve the coordinates of the point located at the intersection of the inertia axis and the 'yz plane'. To do so, create the formulas below:

```
X=intersect(Geometrical Set.1\Line.1, 'yz plane').coord(1)
Y=intersect(Geometrical Set.1\Line.1, 'yz plane').coord(2)
Z=intersect(Geometrical Set.1\Line.1, 'yz plane').coord(3)
```

   You get the intersect function from the Wireframe constructors and the *point*.coord method from the Measures item of the dictionary.

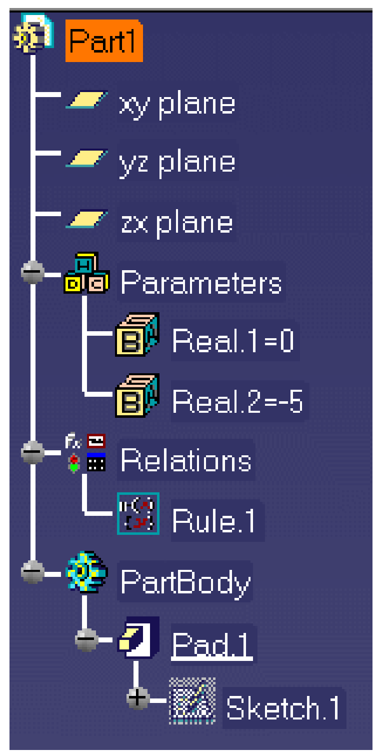
8. Check the value displayed in the specification tree as well as in the Formulas dialog box.

The KwoGettingStarted.CATPart document used as a sample for the Product *Engineering Optimizer User's Guide* illustrates this scenario.

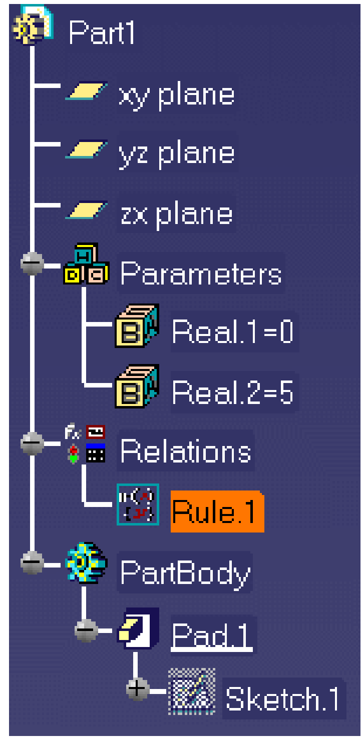
Applying Ranges to Parameters by Using a Rule

This task explains how to apply ranges to parameters by using a rule.

1. Open the KwrRangesParameters.CATPart.
2. Click the icon and select **Real** in the scrolling list to create two parameters of Real type: Real.1 and Real.2.
3. Select Real.1 and right-click the field next to the **Edit name or value of the current parameter** box.
4. Select **Add Range ...** The **Range of Real.1** dialog box opens.
5. Specify the Minimum and the Maximum bounds (-5 and 5 for example), and click **OK** twice.
6. Access the Knowledge Advisor workbench and click the Rule icon ( ). The Rule editor opens.
7. Enter the following rule: **Real.2 =Real.1 .InferiorRange** and click **OK:** Real.2 value changes to -5.



8. Double-click the rule under the Relations node and replace the existing script with **Real.2 =Real.1 .SuperiorRange** and click **OK:** Real.2 value changes to 5.



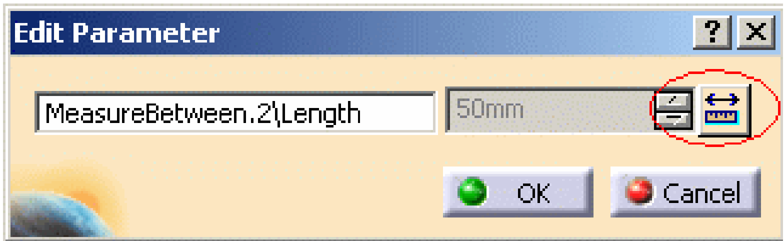
Creating an Associative Link Between Measures and Parameters

This scenario explains how to create a persistent and associative link between a measure created using the **Measure Item** or **Measure Between** command and a parameter.

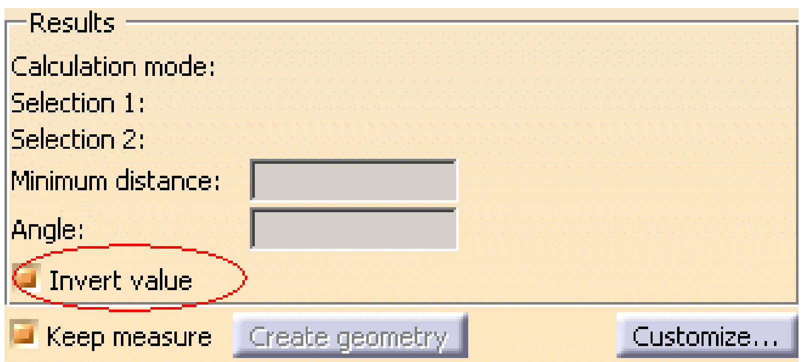
- **Measure Item** allows you to get the length of a curve (edge, line, curve), radius or angle depending on the parameter magnitude.
- **Measure Between** allows you to get the minimal distance or angle between two elements, depending on the parameter magnitude.

This link can be created only if the **Keep measure** option is checked in the **Measure Item** and **Measure Between** dialog boxes (if not the result is copied as a simple value.)

- No formula is created when using the **Measure Item** or the **Measure Between** commands.
- The icon located on the right of the editor field is a measure between or item icon. Note that you will be able to edit the measure.

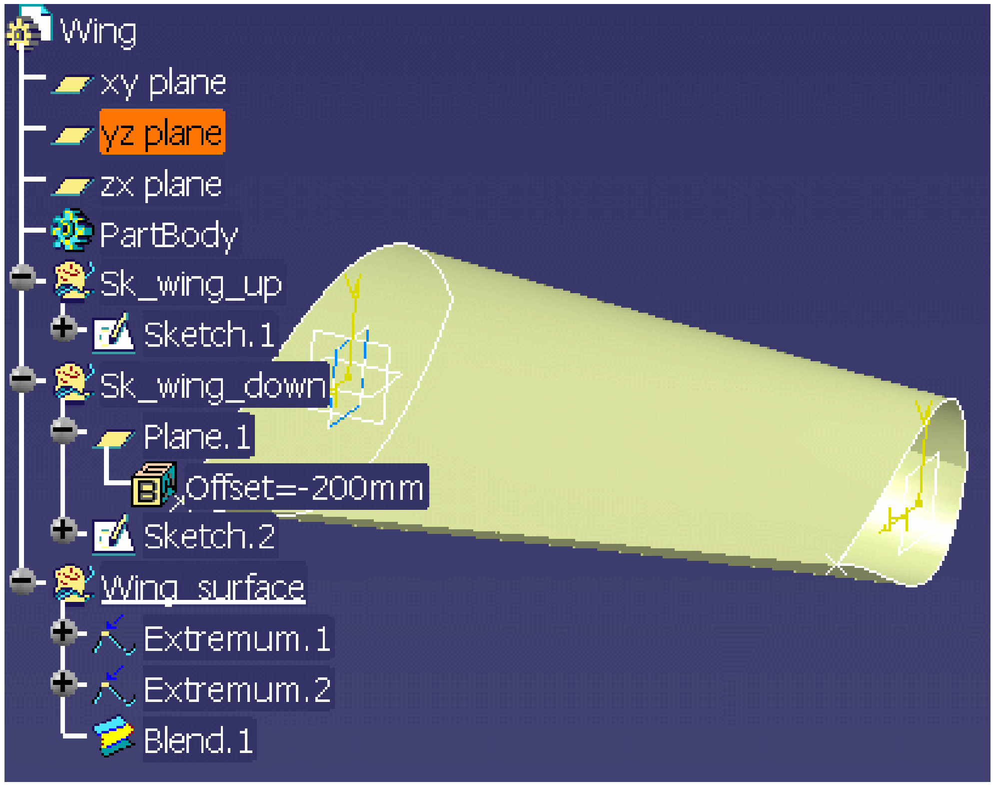


- The parameters located below the Parameters node are directly linked to the measures.
- You can invert the sign of the parameter using the **Invert value** command in the Measure Item or Measure Between panel. The sign concerns only the valuated parameters and not the parameter of the measure.



- To have an associative link, you must make an associative measure. If you select the **Picking point** mode and the **Measure between** function, the measure will not be associative. As a result, there will be no associative geometry.
- When a measure is not associative, the value displays in the value field.
- Even in the case of an associative measure, if you only want to get the result of the measure, uncheck the **Keep measure** check box.
- To create a "smart" customization, click the **Customize...** button in the Measure Item dialog box to see the properties the system can detect for the various types of item you can select.

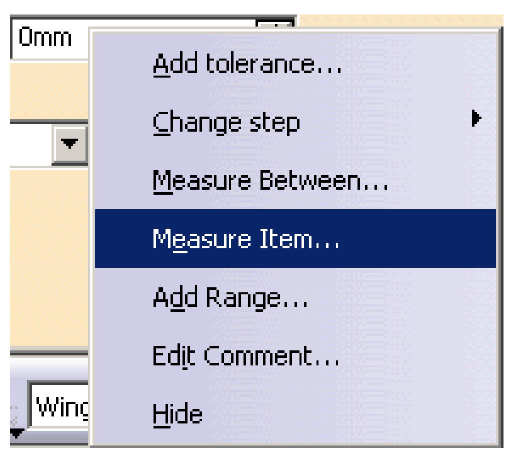
1. Open the KwrPlaneWing.CATPart file. The following image displays.



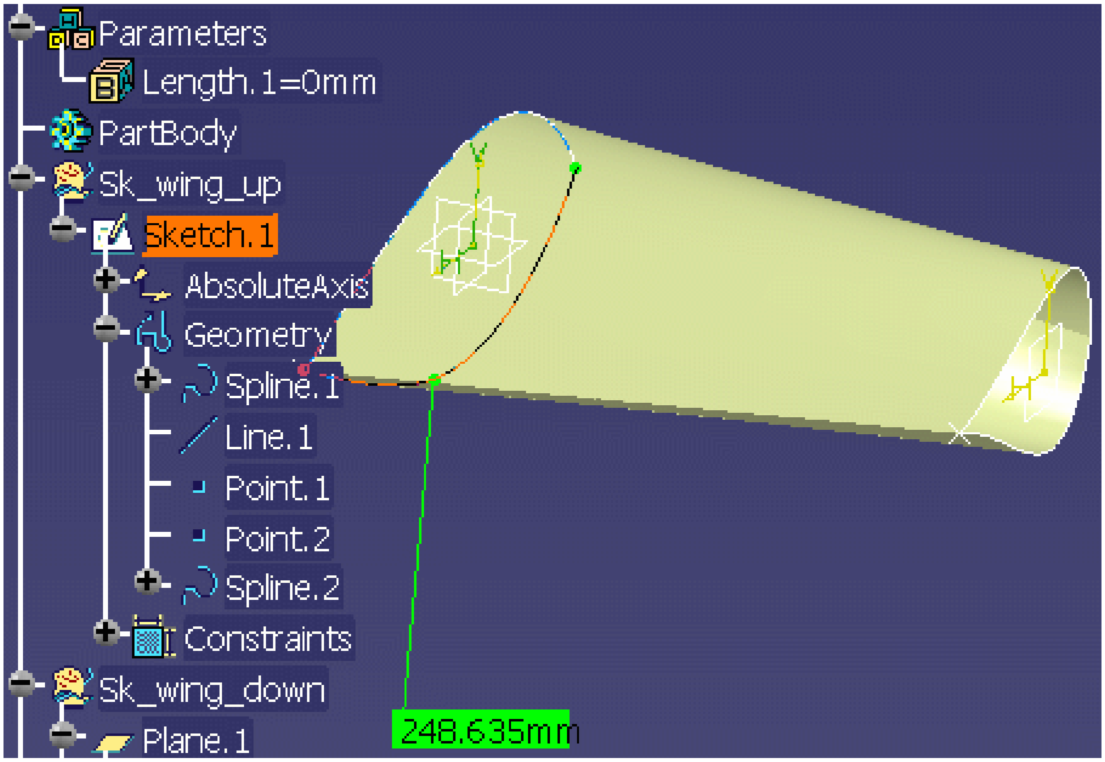
Using the Measure Item... command

2. Add a parameter of **Length** type. To do so, proceed as follows:

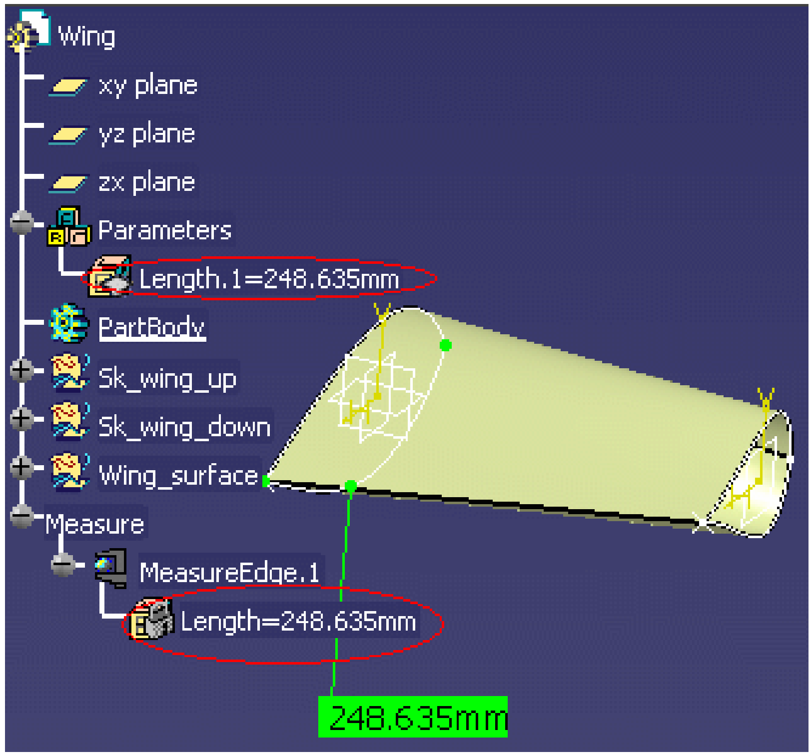
   - Click the Formula icon (). The Formulas dialog box displays.
   - In the **New parameter of type** scrolling list, select **Length** and click the **New parameter of type** button. **Length.1** displays in the **Edit name or value of the current parameter** field.
   - Right-click the value field of **Length.1** and select the **Measure Item...** command. The **Measure Item** dialog box displays.



- Make sure the **Keep measure** option is checked in the **Measure Item** dialog box.
- In the specification tree, expand the Sketch.1 node, and select Spline.2. The selected item is highlighted in the geometry and its measure is displayed in green.

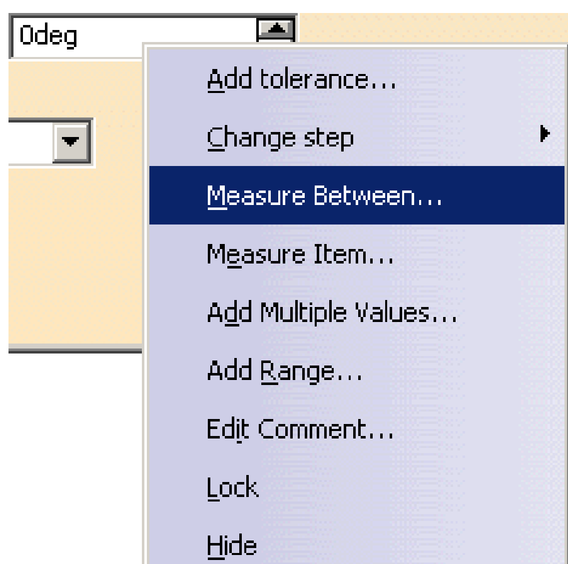


- Click **OK** in the **Measure Item** dialog box and **OK** in the Formulas dialog box. A new parameter is added below the Parameters node and below the Measure node. The Length.1 parameter is now linked to the result of the measure.



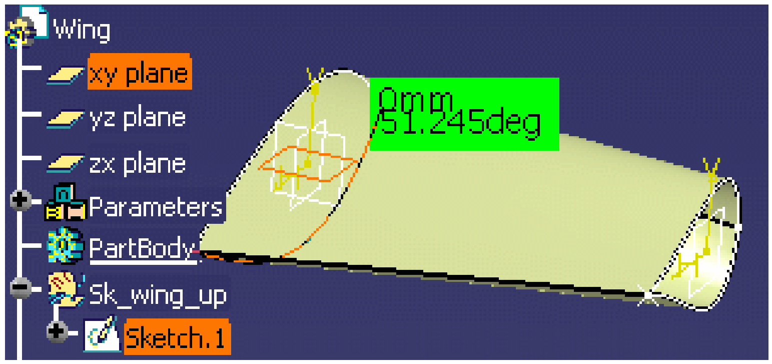
Using the Measure Between... command

1. Add a parameter of **Angle** type. To do so, proceed as follows:
   - Click the Formula icon (). The Formulas dialog box displays.
   - In the **New parameter of type** scrolling list, select **Angle** and click the **New parameter of type** button. **Angle.1** displays in the **Edit name or value of the current parameter** field.
   - Right-click the value field of **Angle.1** and select the **Measure Between...** command. The **Measure Between** dialog box displays.

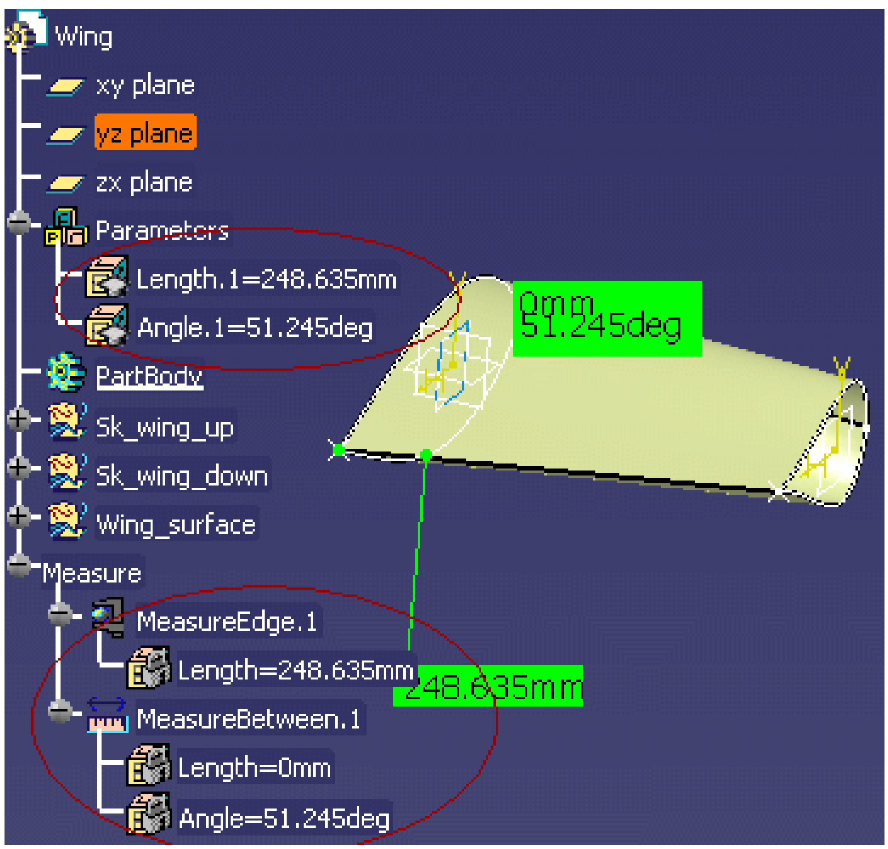


- In the **Selection 2 mode** scrolling list, select the **Edge only** option.
- In the specification tree, select Plane xy then select the geometry as shown below.

The selected items are highlighted in the geometry and the measure is displayed in green.



- Click **OK** in the **Measure Between** dialog box and **OK** in the Formulas dialog box. An angle parameter is added below the Parameters node and the measure displays below the Measure node.



- Click here to display the result sample.

Note that:

- if several characteristics of the measure are computed and have the same magnitude, the system will choose the most convenient according to predefined rules.
- To remove the link to the measure, right-click the measure item in the specification tree and select the **measure object->Remove the link with measure** command.

Publishing Parameters

This scenario explains how to publish parameters. The scenario described below is divided into the following steps:

- Add parameters to the Screw.2 document and publish its Diameter, Depth, and Volume parameters. Repeat the same operations with the second CATPart file.
- Create a CATProduct file and import Screw.2.
- In the context of the Bolt product, insert the Nut part that imports the Depth and the Diameter parameters by selecting the publication MyDepth and MyDiameter of Screw.2.
- In the context of the bolt, replace Screw.2 (KwrScrew.CATPart) by Screw.2 (KwrScrew2.CATPart) that doesn't have the same structure as the first one but owns the same publications. Both the parameters and the check are recomputed.

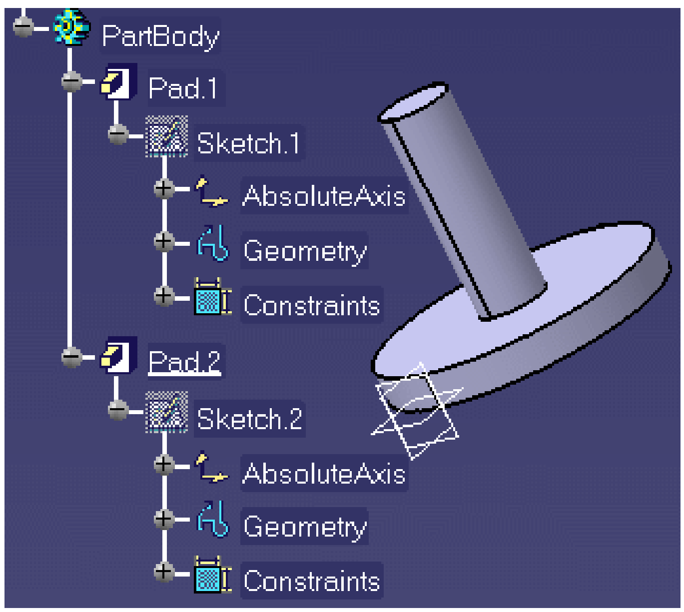
A publication has a name and references a geometry or parameters inside the product (or one of its sub-products).

The publication of parameters should be used when:

- Defining an import of parameters between two parts (similar to the import of geometry).
- Defining relations at the assembly level between parameters (similar to constraints).

Before you start, make sure that the **Keep link with selected object** check box is checked (**Tools->Options->Part Design->General**).

1. Open the KwrScrew.CATPart document. The following image displays.



2. Add parameters to the part. To do so, proceed as follows:

   - Click the icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Volume** and click the **New parameter of type** button.

- In the **Edit name or value of the current parameter** field, enter the name of the parameter: **MyVolume**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.

- Enter the following formula by using the Dictionary: **smartVolume(PartBody\Pad.1) + smartVolume(PartBody\Pad.2)** . Click **OK**, and **Yes**.

- In the **New parameter of type** scrolling list, select **Length** and click the **New parameter of type** button.

- In the **Edit name or value of the current parameter** field, enter the name of the parameter: **MyDepth**. Click **Apply** and click the **Add Formula** button.

- Enter the following formula: **MyDepth=PartBody\Pad.2\FirstLimit\Length** and click **OK**.

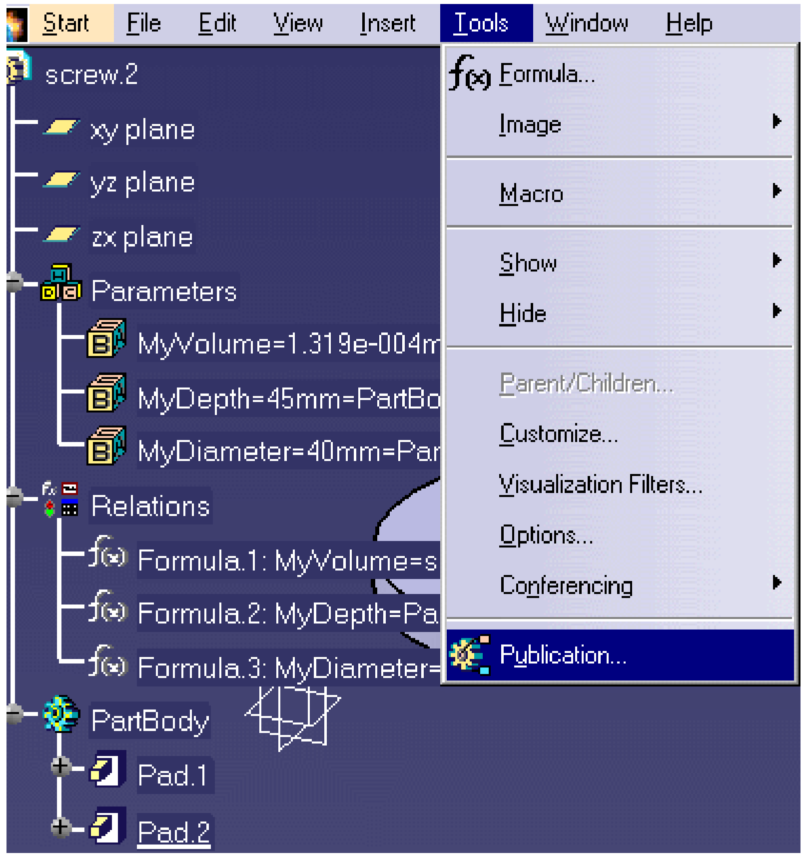
- In the **New parameter of type** scrolling list, select **Length** and click the **New parameter of type** button.

- In the **Edit name or value of the current parameter** field, enter the name of the parameter: **MyDiameter**. Click **Apply** and click the **Add Formula** button.

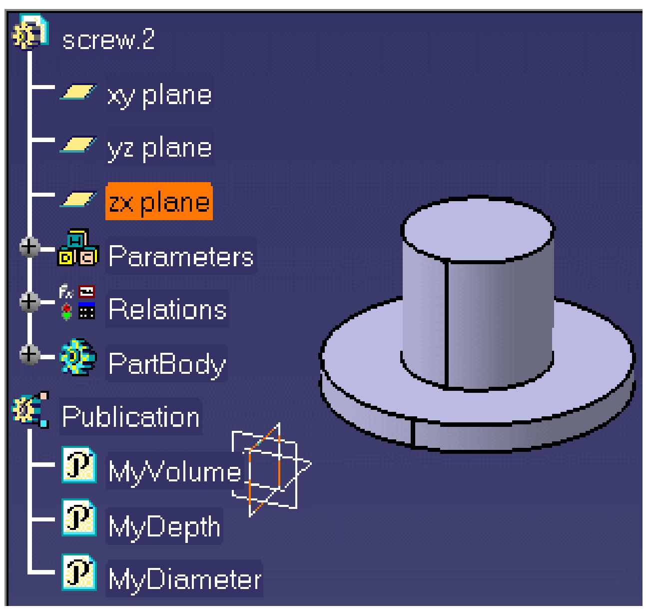
- Enter the following formula: **MyDiameter=PartBody\Sketch.2\Radius.2\Radius * 2**. Click **OK** twice.

3. Publish the **MyVolume**, **MyDepth**, and **MyDiameter** parameters.

   To do so, select the **Tools->Publication** command and select the **MyVolume**, **MyDepth**, and **MyDiameter** parameters in the specifications tree. Click **OK**. The published parameters appear in the specifications tree below the Publication node. Close the file.

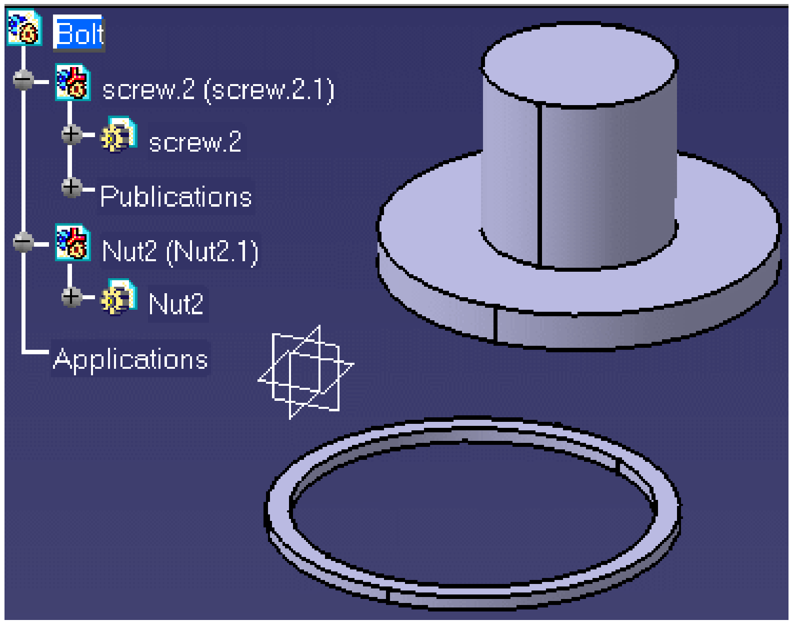


4. Open the KwrScrew2.CATPart and repeat the steps listed above (steps 1 to 3 included). The part should be identical to the one below. Close the file.



5. Create a CATProduct file. Select the **Insert->Existing Component...** command and click the root of the specifications tree. The **File selection** box displays. Select the KwrScrew.CATPart file and click **Open**. The screw is imported.

6. Select the **Insert->Existing Component...** command, select the Kwrnut.CATPart file and click **Open**. The nut part is inserted.



7. Double-click the inner circle of the nut, the Hole Definition window displays.

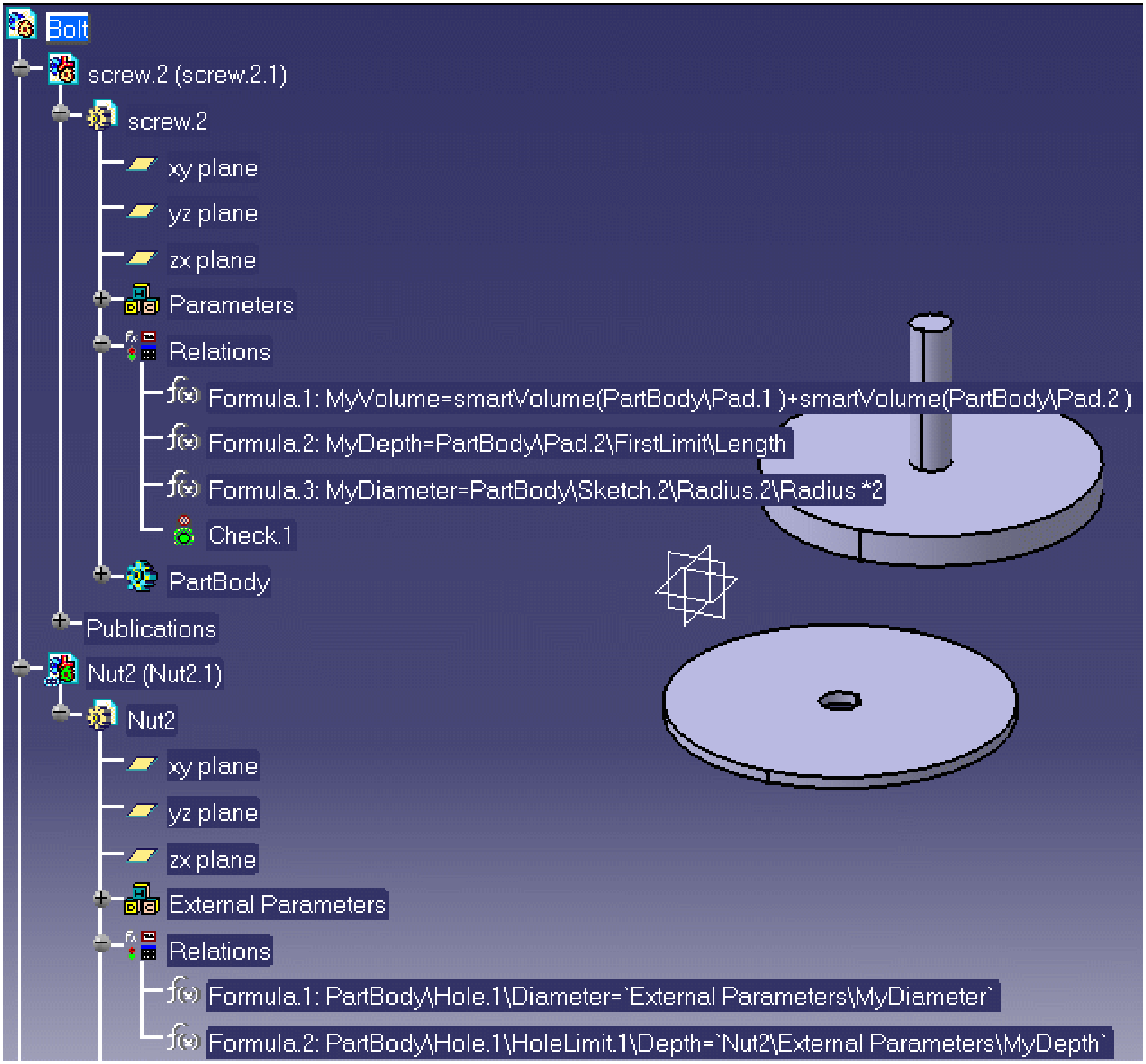
   - Right-click the **Diameter** field and select the **Edit formula...** command. The Formula Editor opens.

   - Select MyDiameter in the screw publications. The formula should be as follows: **PartBody\Hole.1\Diameter=`External Parameters\MyDiameter`**. Click **OK**.

   - Right-click the **Depth** field and select the **Edit formula...** command. The Formula Editor opens.

   - Select MyDepth in the screw publications. The formula should be as follows: **PartBody\Hole.1\HoleLimit.1\Depth=`External Parameters\MyDepth`**. Click **OK** twice.

8. Double-click, then right-click the Screw.2 component in the specifications tree and select the **Components->Replace Component...** command. The **File Selection** window opens. Select the KwrScrew2.CATPart file and click **Open**.
9. Click **Yes** when asked if you want to replace all instances with the same reference as the selected product. Update the nut part: the parameters are recomputed.


Bolt
screw.2 (screw.2.1)
screw.2
xy plane
yz plane
zx plane
Parameters
Relations
Formula.1: MyVolume=smartVolume(PartBody\Pad.1)+smartVolume(PartBody\Pad.2)
Formula.2: MyDepth=PartBody\Pad.2\FirstLimit\Length
Formula.3: MyDiameter=PartBody\Sketch.2\Radius.2\Radius *2
Check.1
PartBody
Publications
Nut2 (Nut2.1)
Nut2
xy plane
yz plane
zx plane
External Parameters
Relations
Formula.1: PartBody\Hole.1\Diameter=`External Parameters\MyDiameter`
Formula.2: PartBody\Hole.1\HoleLimit.1\Depth=`Nut2\External Parameters\MyDepth`

Getting Familiar with the Parameters Explorer

Contrary to the parameters created using the Formulas editor, the parameters created using the Parameters Explorer display below the feature selected in the specification tree.

The Parameters Explorer dialog box is displayed when you click the icon in the standard tool bar. This dialog box allows you to add parameters to features. It is made up of the following fields:

- Feature
- Parameters
- Parameter
- Properties
- Ranges

Feature

This field indicates the item selected in the specification tree to which the parameter will be added.

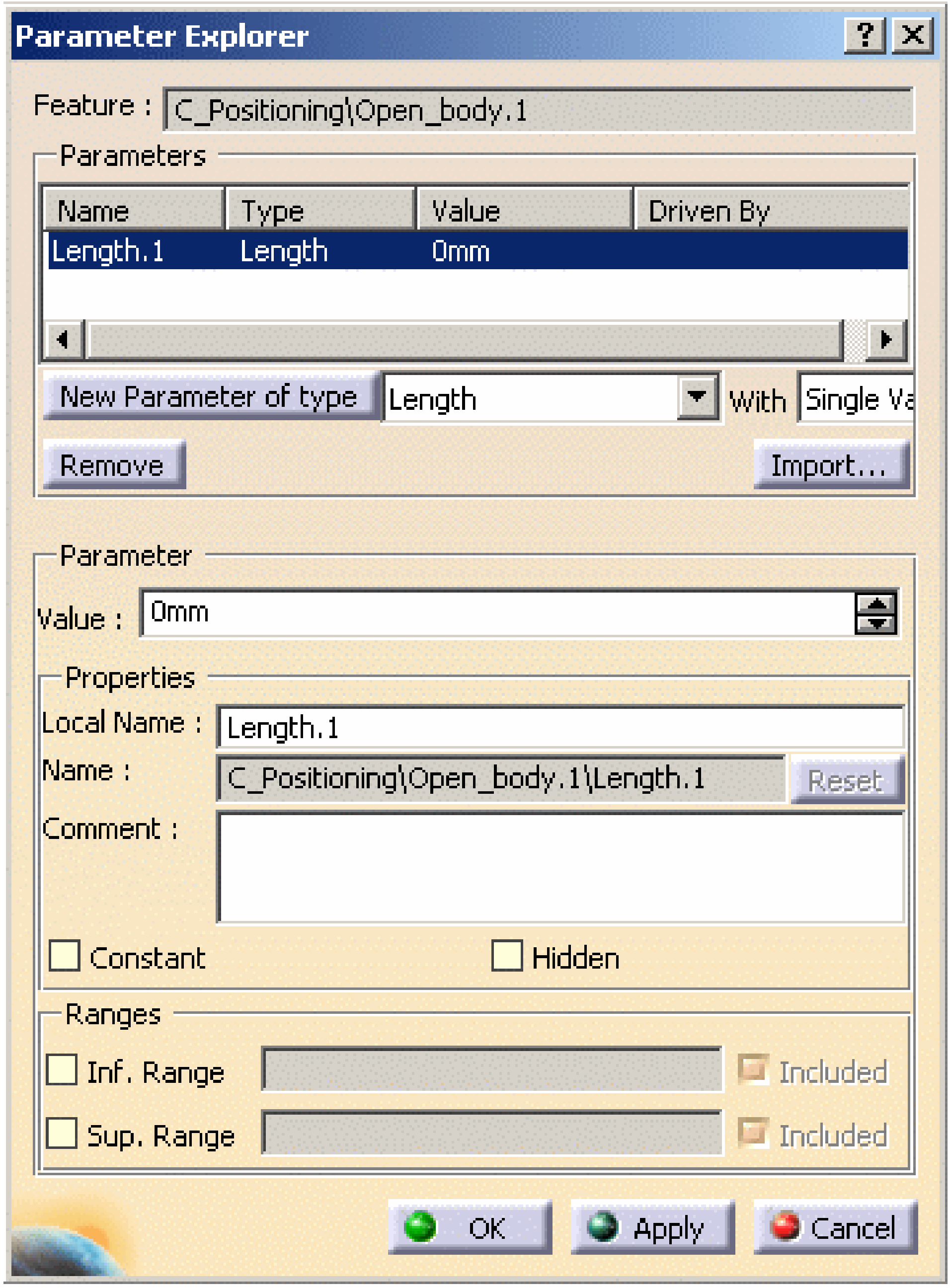
Parameters

This field enables the user to create the parameters that will be added to the feature that he selected in the specification tree. To know more about this field, see Getting Familiar With the ***f(x)*** Dialog Box.

Parameter

The **Value** field enables the user to assign a value to the created parameter.

Properties



The **Local Name** field enables the user to modify the name of the parameter that he created.

The **Name** field indicates the way the parameter will display in the editors.

The **Comment** field enables the user to add comments to the parameter.

The **Constant** check box, if checked, enables the user to lock the parameter. In this case, the parameter cannot be modified.

The **Hidden** check box, if checked, enables the user to decide if the wants the parameter to display or not.

Ranges

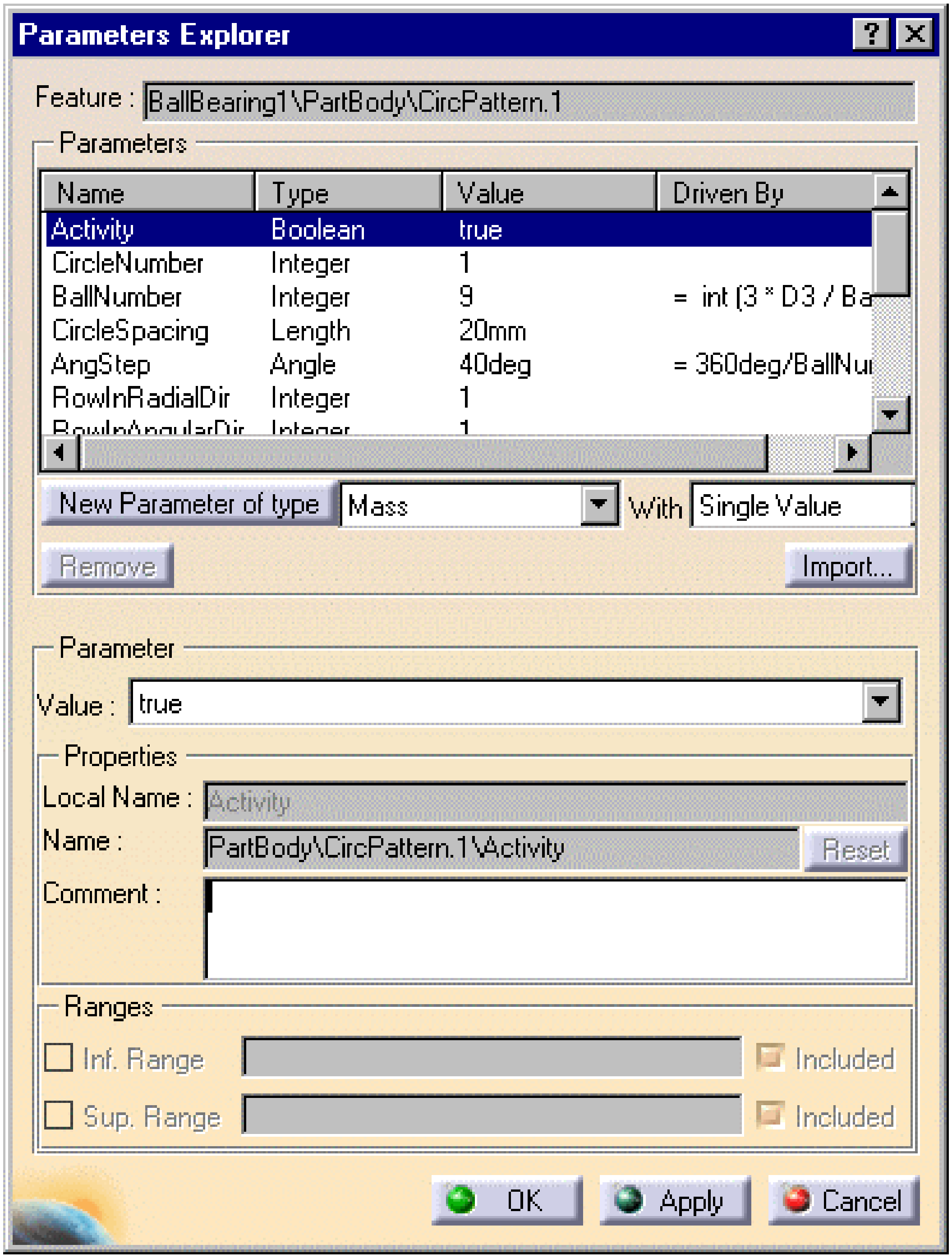
The **Inf. Range** check box, if checked, enables the user to add an inferior range to the parameter.

The **Sup. Range** check box, if checked, enables the user to add an inferior range to the parameter.

Adding a Parameter to a Feature

This task explains how to add two parameters to a circular pattern feature. One parameter is a multiple value string, the other is a mass with upper and lower bounds.

1. Open the KwrBallBearing1.CATPart document.
2. In the specification tree, select the root feature, then select the **Start->Knowledgeware->Knowledge Advisor** command to access the Knowledge Advisor workbench.
3. In the specification tree, select the CircPattern.1 feature.
4. Click the **Parameters Explorer** icon, the dialog box below is displayed:



5. In the **New Parameter of Type** list, select the **String** type, then in the opposite field ('**With**'), select the Multiple Values item. Click **New Parameter of Type**.

6. In the <**Value list** dialog box:

   a) enter the Type1 string, then press **Enter**

   b) enter the Type2 string, then press **Enter**

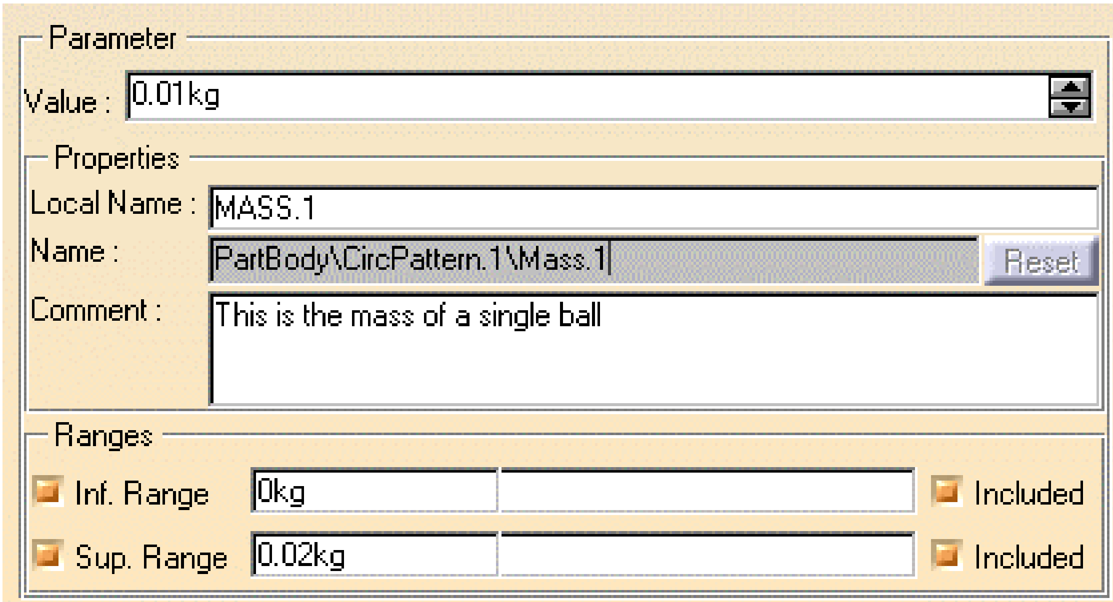
   c) click **OK** to go back to the **Parameter Explorer** dialog box.

7. If need be, rename the created parameter in the **Local Name** field and add a comment.

8. In the **New Parameter of Type** list, select the **Mass** type, then in the opposite field

('With'), select the Single Value item. Click **New Parameter of Type**. The MASS.1 name is displayed by default in the Properties and a default value of 0kg is assigned to the created parameter.

9. Modify these values as indicated on the figure below:



10. Click **OK**. Both parameters are displayed in the specification tree right below the CircPatter.1 feature.

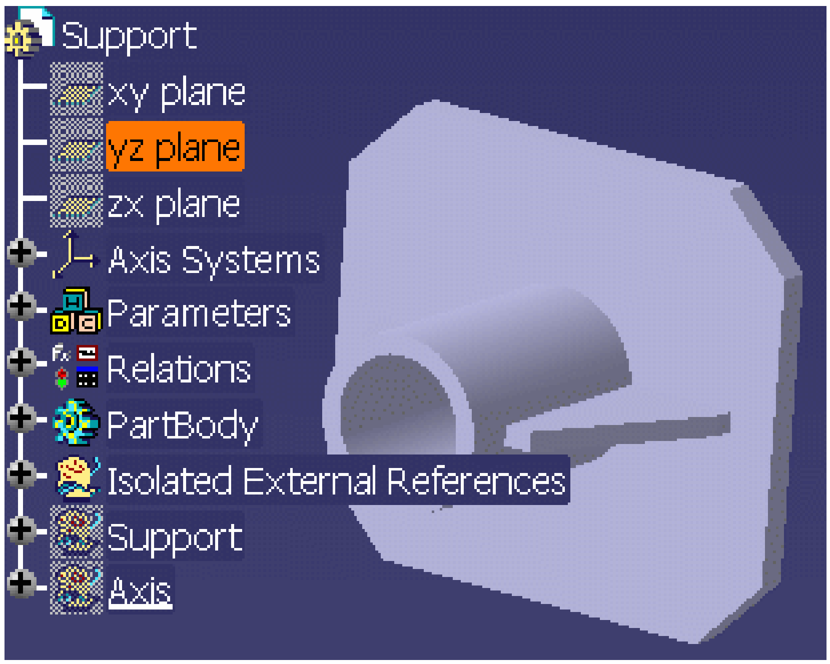
Parameters added by using the Parameters Explorer are displayed right below the feature they are assigned.

Adding a Parameter to an Edge

This task explains how to add parameters to an edge by using the Parameters Explorer.

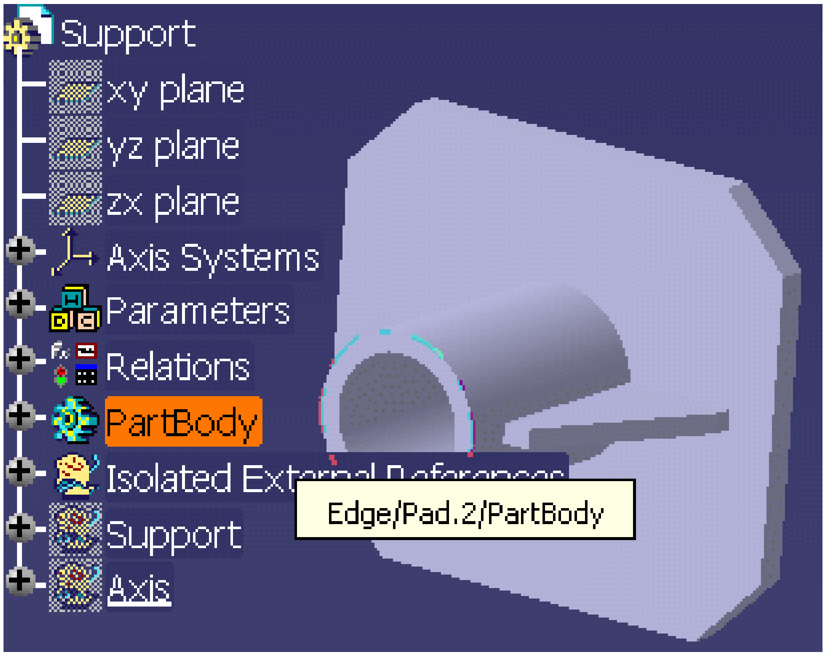
Note that this new function is designed to work on edges, faces and vertex.

1. Open the KwrSupport.CATPart file. The following image displays.



2. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench.

3. Click the **Add Parameters on Geometry** icon () and select the upper edge of Pad.2.

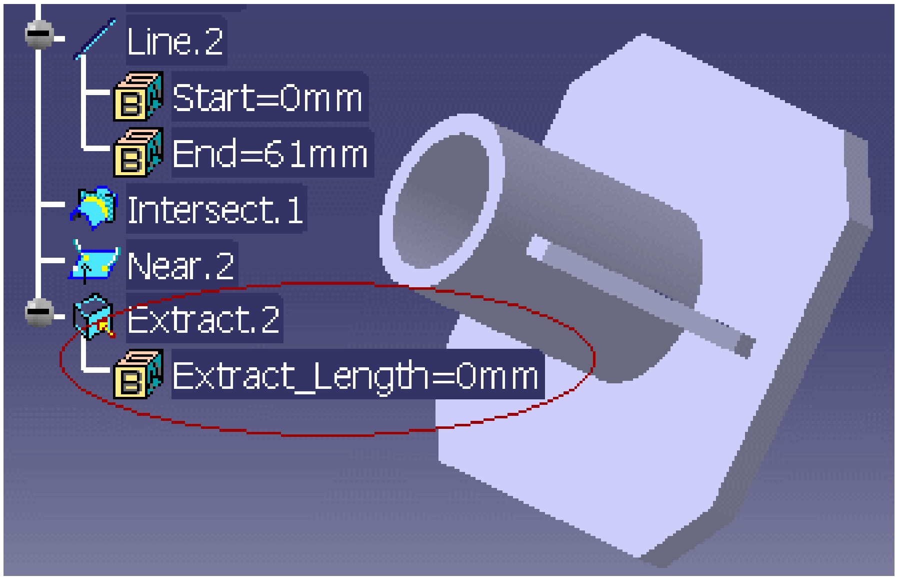


The Parameter Explorer dialog box opens.

4. In the **New Parameter of type** scrolling list, select the **Length** parameter type, and click the **New parameter of type** button.

5. In the **Local Name** field, enter the name of the parameter. For the purpose of this scenario, enter Extract_Length, and click **OK** to validate.

   A new Extract feature is created and the parameter you just created is added to this feature.



Parameters added by using the Parameters Explorer are displayed right below the feature they are assigned.

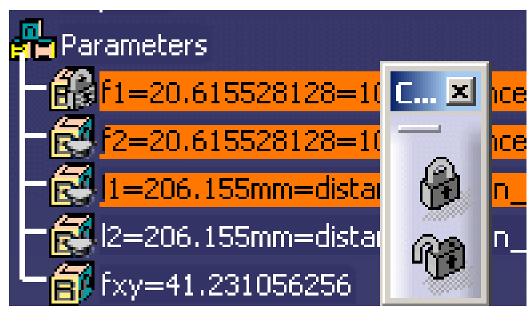
Locking and Unlocking a Parameter

This task explains how to lock and unlock parameters: 2 new commands are now available in the Knowledge toolbar: the **Lock selected parameters...** and the **Unlock selected parameters...** commands.

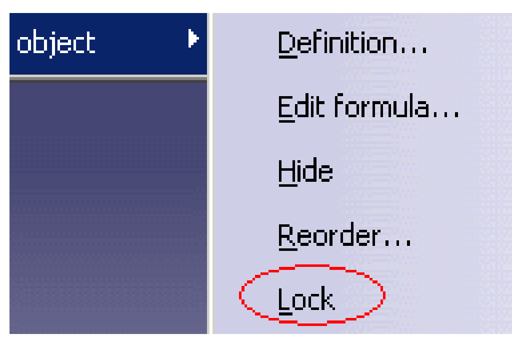
1. Open the KwrLockingUnlockingParameters.CATPart file.
2. Expand the Parameters node and click the f1 parameter.
3. In the **Knowledge** toolbar, click the **Lock selected parameters...** icon ( ) to lock this parameter. The parameter is locked (a lock displays next to the parameter in the specification tree.)
4. To unlock the parameter, select the f1 parameter in the specification tree and click the **Unlock selected parameters...** icon ( ) in the Knowledge toolbar.

Note that:

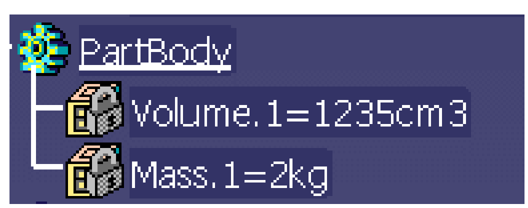
- The **Lock selected parameters...** and **Unlock selected parameters...** commands enable the multiselection of parameters or features.



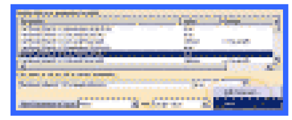
- The parameter contextual menu can be used to lock or unlock the parameter.



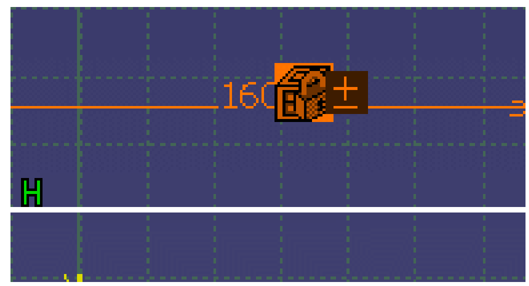
- The **Lock selected parameters...** and **Unlock selected parameters...** commands are recursive: If you select a feature, the parameters located below this feature are locked or unlocked.



- Parameters that do not have a 3D representation can be locked and unlocked. To do so, proceed as follows:
  - Open the Formulas Editor to access the parameter.
  - Select the parameter in the Parameters list
  - In the **Edit name or value of the current parameter** value field, access the contextual menu and select the **Lock/Unlock** command. (click the graphic opposite to enlarge it.)



- Constraints can be locked and unlocked. To do so, proceed as follows:
  - Select the constraint in the specification tree and click the **Lock selected parameters...** () / **Unlock selected parameters** () icon, or
  - Right-click the constraint in the specification tree and select the **Lock the parameter/Unlock the parameter** command.



A lock displays next to the constraint in the Sketcher. Locked constraints display in orange.

Creating Sets of Parameters

This task explains how to create sets of parameters.
You can create sets of parameters below the Parameters node of the specification tree. Using this capability enables you to regroup parameters by categories.

1. Check at least the Parameters and Relations options of the Display tab in the **Tools->Options...->Infrastructure->Part Infrastructure** settings.
2. Open any document containing at least one parameter or create a document and add a parameter to it (otherwise, you won't have the Parameters node displayed in the specification tree).
3. Click the icon, then select the Parameters node in the specification tree. The Parameters.1 (or Parameters.n) parameter set is added to the specification tree right below the Parameters node.
4. Click the icon to add a new parameter in the created parameter set. The Parameter Explorer dialog box is displayed. In the specification tree, select the Parameter Set you want to add a parameter to. The name of the parameter set is displayed in the Feature field of the Parameter Explorer dialog box.
5. Fill in the other fields of the Parameter Explorer dialog box. If need be, see Adding a Parameter to a Feature.
6. After you have finished specifying the new parameter, click OK in the Parameter Explorer dialog box. In the specification tree, you can expand the feature which represents the parameter set. A new parameter has been added below the parameter set.

Parameters belonging to a parameter set can be reordered by using the **Reorder...** command from the contextual menu.

Working with Formulas

Select the Formula icon to specify relations between parameters.

Introducing Formulas

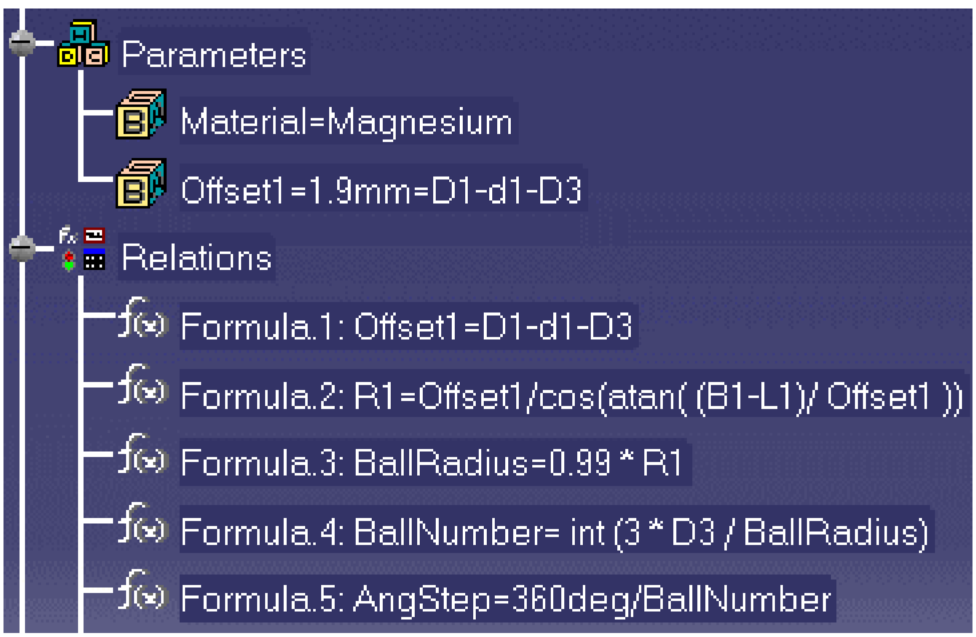
Formulas are features used to define or constrain a parameter. A formula is a relation: the left part of the relation is the parameter to be constrained, the right part is a statement. Once it has been created, a formula can be manipulated like any other feature from its contextual menu. The formula language uses operators and functions of all types whereby you can carry out operations on parameters.

Displaying Formulas in the Specification Tree

Formulas are relations and as such they can be displayed below the Relations node provided you check the 'Relations' box below the 'Specification tree' settings in the **Tools->Options->Infrastructure-> Part Infrastructure->Display** dialog box.

In addition, formulas can also be displayed below the Parameters node provided you check:

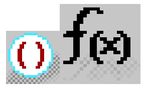
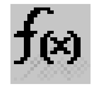
- the 'Parameters' box below the 'Specification tree' settings in the **Tools->Options->Infrastructure-> Part Infrastructure->Display** dialog box
- as well as the 'With Formula' box below the Parameter Tree View settings in the **Tools->Options->General->Parameters and Measure** dialog box



The Activity Parameter

A formula is a feature which is assigned a parameter called the *activity*. The activity value is a boolean. If the activity is set to true, the parameter value cannot be calculated from the formula. If a formula is created for a parameter which is not already constrained by another formula, the activity of the new formula is set to true by default.

A parameter can be constrained by several formulas, but only one formula can be active at a time. Before activating a formula on a given parameter, you must deactivate the other formulas defined on the same parameter.

Activity value	false	true
Relation icon in the specification tree		

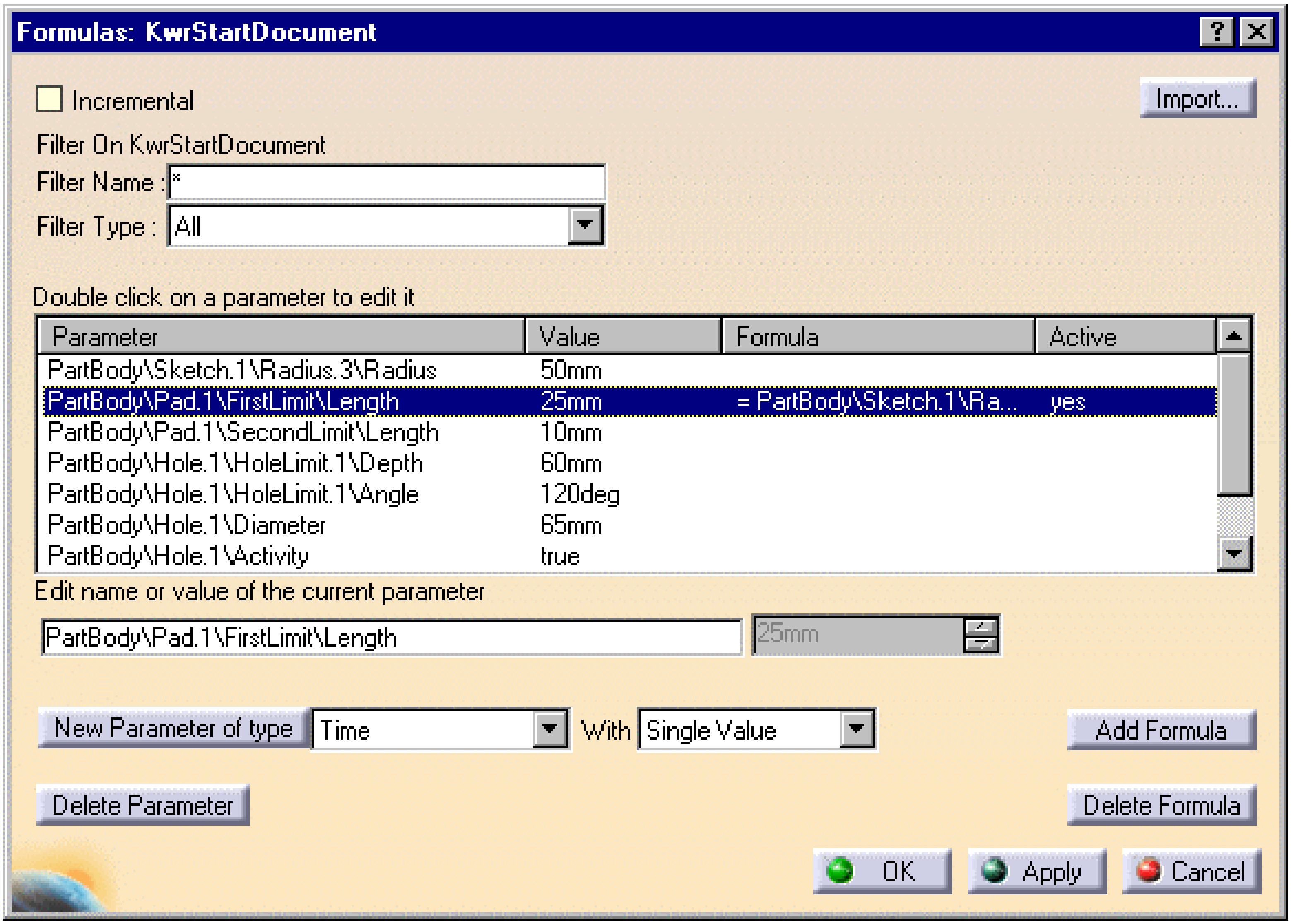
Importing Formulas

Parameters as well as the associated formulas can be imported from an external file. Refer to Introducing Parameters and Importing Parameters for more information on how to import formulas.

Getting Familiar With the *f(x)* Dialog Box

The *f(x)* dialog box is displayed when you click the  icon in the standard tool bar. This dialog box allows you to:

- Display the list of parameters
- Create parameters and formulas
- Import external files.



The parameter list

Basically, the parameter list displays the parameters related to the feature selected either in the specification tree or in the geometry area. If no feature has been selected, all the document parameters are displayed. The dialog box being open, you can select a given feature either in the tree or in the geometry area and display its related parameters.

You can restrict the list of displayed parameters by using the Filter Name and Filter Type capabilities as well as the Incremental check box.

The Filter Name filter

This filter allows you to narrow the list of displayed parameters by specifying a substring. If you specify *Limit* as filter, only the parameter with Limit as sub-string will be displayed, for example:

PartBody\Pad.1\FirstLimit\Length
PartBody\Pad.1\SecondLimit\Length
PartBody\Hole.1\HoleLimit.1\Depth
PartBody\Hole.1\HoleLimit.1\Angle

The Filter Type filter

This filter allows you to restrict the list of parameters by specifying a type. Selecting User parameters will display only the parameters created by the New Parameter of type button. Selecting Hidden parameters will display only the list of parameters which have been declared as hidden by using the Hide command from the value field contextual menu.

The Hide command is only available for user parameters.

The Incremental check box

Selecting a feature in the specification tree or in the geometry area displays in the editor only the first level of features right below the selected feature. The parameter list on figure above displays all the parameters related to the Pad.1 and Hole.1 features. Selecting Pad.1 in the tree (Incremental unchecked) will display the parameters below:

PartBody\Pad.1\FirstLimit\Length
PartBody\Pad.1\SecondLimit\Length
PartBody\Sketch.1\Radius.3\Radius

Checking Incremental restricts the list of parameters to the one below:

PartBody\Pad.1\FirstLimit\Length
PartBody\Pad.1\SecondLimit\Length

The 'Edit name of value of the current parameter' field

This field displays the parameter which has been selected in the parameter list. The value field on the right-hand side is grayed out when the parameter is constrained by a formula, a design table or any type of relations. Right-clicking this value field provides you with a number of commands whereby you can refine the parameter definition.

The New Parameter of type button

This button allows you to create a user parameter. This user parameter can be assigned a single value or multiple values (akin to the enum idea).

The Delete Parameter button

This capability operates only for user parameters.

The Add Formula button

When you create a formula, you specify that a parameter, whatever its type, is to be constrained by a relation. Clicking the **Add Formula** button displays the Formula editor. The formula which is created is displayed in the parameters list as well as its activity.

To know more about the Dictionary available in the Formula editor, see Using the Dictionary.

The Delete Formula button

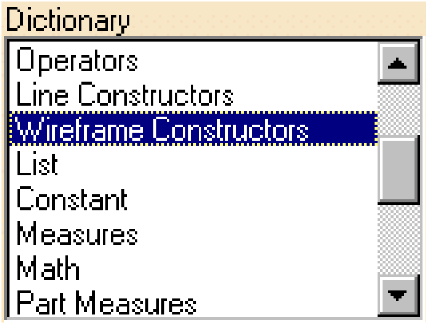
When a parameter which is constrained by a formula is selected in the parameter list, clicking Delete Formula removes the formula.

The Import button

This capability allows you to import parameters and parameter values from a text file or from an Excel file (Windows).

Using the Dictionary

The Dictionary allows you to access the functions, operators and feature attributes that can be used in relations.
It can be accessed both from the Formula Editor, the Rule Editor as well as from the Check Editor.



Packages displayed in the left part of the browser are those you selected from the **Tools->Options ->General->Parameters and Measure->Language** tab.

Design tables	Operators	Point Constructors
Law	Line Constructors	Circle Constructors
String	Direction Constructors	List
Measures	Surface Constructors	Wireframe Constructors
Part Measures	Plane Constructors	Analysis Operators
Math	Electrical Functions	

Constants

The following constants are specified or recognized by *CATIA* when programming rules and checks. As a result, they can be used anywhere in a relation in place of the actual values.

- false - one of the two values that a parameter of type Boolean can have
- true - one of the two values that a parameter of type Boolean can have
- PI - **3.14159265358979323846 -** The ratio of the circumference of a circle to its diameter.
- E - The base of natural logarithm - The constant *e* is approximately **2.718282**.

Design Table Methods

CloserSupConfig Method

Applies to a design table sheet. Returns the configuration which contains the smallest values greater or equal to the values of the given arguments. When several configurations meet this condition, the method sorts out the possible configurations with respect to the column order as it is specified in the argument list.

Syntax

sheet.**CloserSupConfig**(*columnName*: String, *minValue*: Literal, ...): Integer

The **CloserSupConfig** function takes the following arguments:

Arguments	***Description***
columnName	Should be put in quotes. At least, one couple of arguments *columnNamei*/ *minValuei* is required
minValue	Required. You should specify the units.

Example

Given the design table below:

	SketchRadius(mm)	PadLim1(mm)	PadLim2(mm)
1	120	60	10
2	130	50	30
3	120	60	25
4	140	50	40

The expression below:

Relations\DesignTable1\sheet_name.CloserSupConfig("PadLim1", 60mm, "SketchRadius", 120mm, "PadLim2", 20mm)

returns 3

CloserInfConfig Method

Applies to a design table sheet. Returns the configuration which contains the largest values less or equal to the values of the given arguments. When several configurations meet this condition, the method sorts out the possible configurations with respect to the column order as it is specified in the argument list.

Syntax

sheet.**CloserInfConfig**(*columnName*: String, *maxValue*: Literal, ...):Integer

The **CloserInfConfig** method takes the following arguments:

Arguments	***Description***
columnName	Should be put in quotes. At least, one couple *columnName/maxValue* is required
maxValue	Required. You should specify the units.

Example

Given the design table below:

	SketchRadius(mm)	PadLim1(mm)	PadLim2(mm)
1	120	60	10
2	130	50	30
3	120	60	20
4	140	50	40

The statement below
Relations\DesignTable1\sheet_name.CloserInfConfig("PadLim1", 60mm, "SketchRadius", 130mm, "PadLim2", 40mm)
returns 3.

Explanations

The values of lines 1 , 2 and 3 are all less or equal to the values specified in the method arguments.

- As the first parameter specified in the argument list is "PadLim1", the method scans the lines 1, 2 and 3 and searches for the largest "PadLim1" value which is less or equal to 60 mm. Two configurations meet the condition: configuration 1 and configuration 3.
- As the second parameter specified is "SketchRadius", the method scans the configurations 1 and 3 and searches for the largest "SketchRadius" value less or equal to 130 mm. Again, the function finds two configurations meeting the criteria.
- Then it rescans lines 1 and 3 and searches for the largest "PadLim2" value less or equal to 40mm. The result is line 3.

CloserValueSupInColumn Method

Applies to a design table sheet. Scans the values of a column and returns the greatest cell value which is the nearest to a specified one. Returns 0 if no value is found or if the method arguments are not properly specified.

Syntax

sheet.**CloserValueSupInColumn(***columnIndex:* Integer, *Value:* Real**)**

The **CloserValueSupInColumn** method takes two arguments:

Arguments	***Description***
columnIndex	Required. Index of the table column. Integer from 1 to n.
Value	Required. Value searched for. Should be a real.

Example

```
ValueSup=Relations\DesignTable1\sheet_name.CloserValueSupInColumn(1, 80mm)
Message("Closest sup value is # (0.08 is expected)", ValueSup)
```

CloserValueInfInColumn Method

Applies to a design table sheet. Scans the values of a column and returns the smallest cell value which is the nearest to a specified one. Returns 0 if no value is found or if the method arguments are not properly specified.

Syntax

sheet.**CloserValueInfInColumn**(*columnIndex*: Integer, *value*: Real): Real

The **CloserValueInfInColumn** function has two arguments:

Arguments	***Description***
columnIndex	Required. Number or index of the table column. Integer from 1 to n.
value	Required. Value searched for. Should be a real.

Example

```
Message("Closest inf value is # ", Relations\DesignTable1\sheet_name.CloserValueInfInColumn(2,41mm))
```

MinInColumn Method

Applies to a design table sheet. Returns the smallest of a column values. Returns 0 if the column specified is out of range.

Syntax

sheet.**MinInColumn**(*columnIndex* : Index): Real

where *columnIndex* is the column number.

Example

```
MinimumValue=MinInColumn(3)
Message("Minimum value is # (0  is expected)", MinimumValue)
/* you can use also */
Message("Minimum value is # (0 is expected)", MinInColumn(3))
```

Sample

KwrProgramDT.CATPart

MaxInColumn Method

Applies to a design table sheet. Returns the greatest of a column values. Returns 0 if the column does not contain numerical values or if the method arguments are not properly specified.

Syntax

sheet.**MaxInColumn**(*columnIndex*: Integer): Real

Example

```
MaximumValue=Relations\DesignTable1\sheet_name.MaxInColumn(1)
Message("Maximum value is # (0.150 is expected)", MaximumValue)
```

LocateInColumn Method

Applies to a design table sheet. Returns the index of the first row which contains a specified value. Returns zero if the value is not found or if the method arguments are not properly specified.

Syntax

sheet.**LocateInColumn**(*columnIndex*: Integer, *value*: Literal) : Integer

The **LocateInColumn** method has two arguments:

Arguments	***Description***
ColumnNumber	Required. Number or index of the table column. Integer from 1 to n.
Value	Required. Value searched for. Can be a string or a boolean

Example

```
Line=Relations\DesignTable1\sheet_name.LocateInColumn(4,11mm)
if (Line == 0)
{
Message("No value found !!!")
}
```

CellAsString Method

Applies to a design table sheet. Returns the contents of a cell located in a column. Returns an empty string if the cell is empty or if the method arguments are not properly specified.

Syntax

sheet.**CellAsString(***rowIndex:* Integer**, c***olumnIndex:* Integer**):** String

where *rowIndex* is the configuration number and **c***olumnIndex* the column number.

Example

```
CString=Relations\DesignTable1\sheet_name.CellAsString(1,5)
if (CString == "")
 {
  Message("No value read !!!")
 }
```

CellAsBoolean Method

Applies to a design table sheet. Returns the contents of a cell located in a column intended for boolean values. Returns false if the cell does not contain a boolean or if the method arguments are not properly specified.

Syntax

sheet.**CellAsBoolean**(*rowIndex*: Integer, *columnIndex*: Integer): Boolean

The **CellAsBoolean** method has two arguments:

Arguments	***Description***
rowIndex	Required. Configuration number. Integer from 1 to n.
columnIndex	Required. Index of the table column. Integer from 1 to n.

Example

```
Boolean2=Relations\DesignTable1\sheet_name.CellAsBoolean(1,5)
if (Boolean2 <> true)
{
 Message("Error !!!")
}
```

CellAsReal Method

Applies to a design table sheet. Returns the contents of a cell located in a column intended for real values. Returns zero if the cell does not contain a real or if the method arguments are not properly specified.

Syntax

sheet.**CellAsReal**(*rowIndex*: Integer, *columnIndex*: Integer): Real

where *rowIndex* is the configuration number (integer from 1 to n) and *columnIndex* the column number.

SetCell Method

Enables the user to add a cell at a given position in an Excel file or a tab file.

Note: the index should start at 1 for the (1,1) cell to be located at the left top corner.

Syntax

sheet.**SetCell(***IndexRow:Integer, IndexColumn:Integer, CellValue:Literal***):** Void

LocateInRow

Applies to a design table sheet. Returns the index of the first row which contains a specified value. Returns zero if the value is not found or if the method arguments are not properly specified.

Syntax

sheet.**LocateInRow**(*rowIndex*: Integer, *value*: Literal) : Integer

The **LocateInRow** method has two arguments:

Arguments	***Description***
RowNumber	Required. Number or index of the table row. Integer from 1 to n.
Value	Required. Value searched for. Can be a string or a boolean

Sheet.CloserConfig(string columnName1, string sortMethod1, any value1, columnName2, sortMethod2, value2, ...)

Method enabling the user to find the closest configuration of a design table according to criteria that mix greater than (or equal to) and smaller than (or equal to).

Where:

- The columnName<index> arguments match the columns names in the design table source file.
- The sortMethod<index> are strings that can only take the following values : "<", "<=", ">", ">=", "==", "!="
- The value<index> arguments can be either numerical values (the magnitude should correspond to the magnitude of a given column), or string values. If string values are used, the exact value is retrieved from the cells (the sortMethod is used only if it is "==" or "!=".) If sortmethod is not "==" or "!=", "==" is assumed.

Example:

Sheet.CloserConfig("column1", "<", 10mm, "column2", ">=", 20deg, "column3", "!=", "standard") that can be interpreted as follows:
Find the configuration where:

- the value in column1 is strictly smaller than 10mm,
- where the value in column2 is greater than or equal to 20deg
- where the value in column3 is different from the standard string.

Note that:

- If several configurations are valid, you will find the configuration whose values are as close as possible to the "bounds". If still several configurations are valid, the first valid one is returned.

- If no configuration is valid, the returned value is 0.

Operators

Arithmetic operators

+ Addition operator (also concatenates strings)

- Subtraction operator

* Multiplication operator

/ Division operator

( ) Parentheses (used to group operands in expressions)

= Assignment operator

** Exponentiation operator

Logical Operators

and Logical conjunction on two expressions

or Logical disjunction on two expressions

Comparison Operators

<> Not equal to

== Equal to

>= Greater or equal to

<= Less than or equal to

< Less than

> Greater than

Point Constructors

Sample: KwrPointConstructors

- **point** (*x*: Length, *y*: Length, *z*: Length): Point
  Creates a point from its three coordinates. Values or parameter names can be used to pass the arguments.

Examples:
Specifying values:
Geometrical Set.1\Point.1 =
point(10mm,10mm,10mm)
Specifying parameter names:
Geometrical Set.1\Point.4 =
point(0mm,L3,L1)

- **pointbetween**(*pt1*: Point, *pt2*: Point, *ratio*: Real, *orientation*: Boolean) : Point
  Creates a point between another two points. If true is specified in the fourth argument, the third parameter is the ratio of the distance pt1-new point to the pt1-pt2 distance. If false is specified in the fourth argument, the ratio expresses the distance pt2-new point to the pt1-pt2 distance (to create a point at the middle between pt1 and pt2, specify a ratio of 0.5).

Example:
Geometrical Set.1\Point.5 =
pointbetween(Geometrical Set.1\Point.1, Geometrical Set.1\Point.2, 0.6, true)

- **pointoncurve**(*crv*:Curve, *pt*:Point, *distance*:Length, *orientation*: Boolean) : Point
  Creates a point on a curve. The point is to be created at a given curvilign *distance* from a reference point specified in the second argument. The boolean specified in the fourth argument allows you to reverse the direction in which the point is to be created. If the point specified in the second argument is not on the curve, the projection of this point onto the curve becomes the actual reference point.

Example:
Geometrical Set.1\Point.6 =
pointoncurve(Geometrical Set.1\Spline.1, Geometrical Set.1\Point.5, 5mm, true)

- **pointoncurveRatio**(*crv*:Curve, *pt*:Point, *ratio*:Real, *orientation*: Boolean) : Point
  Creates a point on a curve. The location of the point to be created is determined by the real which is specified in the third argument. This real is the ratio of the distance [point to be created->reference point] to the distance [point to be created->curve extremity]. The boolean specified in the fourth argument allows you to reverse the direction in which the point is to be created. If the point specified in the second argument is not on the curve, the projection of this point onto the curve becomes the actual reference point.

Example:
Geometrical Set.1\Point.7 =
pointoncurveRatio(Geometrical Set.1\Spline.1,Geometrical Set.1\Point.3, 0.4,true)

- **pointonplane**(*pln*:Plane, *pt*:Point, *dx*:Length, *dy*:Length): Point
  Creates a point on plane. The location of the point to be created on the plane is determined by the coordinates (H,V system) passed in the third and fourth arguments. These values are specified with respect to the reference point passed in the second argument.

Example:
Geometrical Set.1\Point.8 =
pointonplane(Geometrical Set.1\Plane.1,Geometrical Set.1\Point.1, 10mm,10mm)

- **pointonsurface**(*sur*:Surface, *Pt*:Point, *Dir*:Direction, *dist*:Length): Point
  Creates a point on surface. The location of the point to be created on the surface is determined by its distance (fourth argument) to a reference point (second argument) along a direction (third argument).

Example:
Geometrical Set.1\Point.9 =
pointonsurface(Geometrical Set.1\Extrude.1,Geometrical Set.1\Point.3,
direction(Geometrical Set.1\Line.1),10mm)

- **center**(circle): Point
  Creates a point from a circle. The circle can be of any type (sketch or GSM circle). The point which is created is the circle center.

Example:
Geometrical Set.1\Point.10 =
circle(Geometrical Set.1\Circle.1)

- **pointtangent**(curve,direction): Point
  Creates the tangency point between a curve and a direction.

Example:
Geometrical Set.1\Point.11 =
pointtangent(Geometrical Set.1\Spline.1, direction(`yz plane`))

- **centerofgravity**(Body): Point
  Constructs the center of gravity of a solid (i.e. a PartBody type feature).

Example:
Geometrical Set.1\Point.12 =
centerofgravity(PartBody)

- **curvaturecenter**(crv: Curve, pt: Point): Point
  Constructs the curvature center of a curve for a given point.

Example:
Geometrical Set.1\Point.13 =
curvaturecenter(Geometrical Set.1\Circle.1, Geometrical Set.1\Point.6)

- **extremum**(Curve, Direction, Boolean, Direction, Boolean, Direction, Boolean)
  Constructs an extremum point. The inputs are a curve, 3 directions, and 3 booleans.

Example:

```
Geometrical Set.1\Point.2=
extremum(`Geometrical Set.1\Circle.1` ,direction(`xy plane` ) ,FALSE,direction(`xy plane`
),TRUE,direction(`xy plane` ),TRUE)
```

- **extremum**(Surface, Direction, Boolean, Direction, Boolean, Direction, Boolean)
  Constructs an extremum. The inputs are a surface, 3 directions, and 3 booleans.

Evaluate Method

Allows you to compute a law whether a KnowledgeAdvisor or a Generative Shape Design Law and use the resulting data within another law.

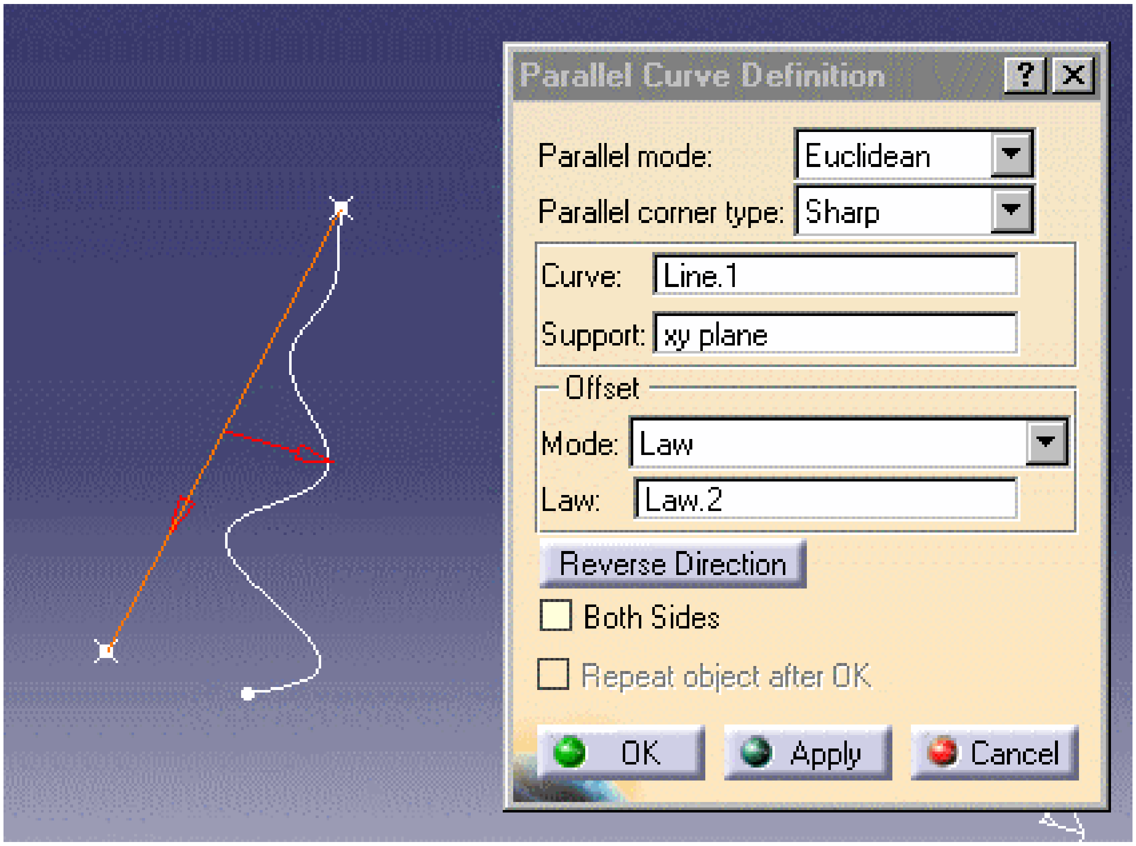
Syntax

law.**Evaluate(***Real***):** Real

where the argument is the parameter to which the law is applied.

Example

1. Create a Generative Shape Design line.
2. Create a first law by clicking the  icon in the standard tool bar.
3. In the law editor, create two real formal parameters.
4. Enter the law (Law.1) below into the editor:
   FormalReal.1 = 5*sin(5*PI*1rad*FormalReal.2)+ 10
5. Click OK to add the law to the document.
6. Repeat the same operation and enter the law (Law.2) below:
   FormalReal.1 = 3* FormalReal.2*Relations\Law.1.Evaluate(FormalReal.2)
7. In the Generative Shape Design workbench, create a line parallel to the line created in step 1. Specify the law which is defined just above in the Offset field.



Sample

KwrObject.CATPart

Line Constructors

Sample: KwrLineConstructors

- **line**(*Point*, *Point*): Line
  Creates a line from two points.

Example:
Geometrical Set.1\Line.1 =
line(Geometrical Set.1\Point.1, Geometrical Set.1\Point.2)

- **line**(*pt*: Point, *dir*: Direction, *start*: Length, *end*: Length, *orientation*: Boolean) : Line
  Creates a line passing through a point and parallel to a direction.
  The third and fourth arguments are used to specify the start and end points.
  The last argument allows you to reverse the line direction.

Example:
Geometrical Set.1\Line.2 =
line(Geometrical Set.1\Point.2, direction(`zx plane`), 0mm, 20mm, true)

- **lineangle**(*crv*: Curve, *sur*: Surface, *pt*: Point, *geodesic*: Boolean, *start*: Length, *end*: Length, *angle*: Angle, *orientation*: Boolean) : Line
  Creates a line passing through a point, tangent to a surface and making a given angle with a curve. When the geodesic argument is set to true, a geodesic line is created(projected) onto the surface.

Example:
Geometrical Set.1\Line.3 =
lineangle(Geometrical Set.1\Spline.1 , Geometrical Set.1\Extrude.1 , Geometrical Set.1\Point.4 , false, 0mm , 50mm , 80deg , false)

- **linetangent**(*crv*: Curve, *pt*:Point, *start*:Length, *end*:Length, *orientation*:Boolean) : Line
  Creates a line tangent to curve at a given point.

Example:
Geometrical Set.1\Line.5 =
linetangent(Geometrical Set.1\Spline.1, Geometrical Set.1\Point.6 ,0mm, 30mm, true)

- **linenormal**(*sur*:Surface, *pt*:Point, *start*:Length, *end*:Length, *orientation*:Boolean) : Line
  Creates a line normal to a surface at a given point.

- **mainnormal**(*crv*: Curve, *pt*: Point) : Line
  Creates a line normal to a curve at a given point.
  The line is created in the plane which contains the tangent vector.

- **binormal**(*crv*: Curve, *pt*: Point) : Line
  Creates a line normal to a curve at a given point.
  The line is created in plane which is orthogonal to the tangent vector.

- **InertiaAxis**(rank: Integer, Body, ...):Line
  Enables to determine the inertia axis of a body.

Example:
Geometrical Set.1\Line.1=
inertiaAxis(1,PartBody\Pad.1)

Circle Constructors

Sample: KwrCircleConstructors.CATPart

- **circleCtrRadius** (*center*: Point, *support*: Surface, *radius*: Length, *limits*: Integer, *start*: Angle, *end*: Angle): Circle

Creates a circular arc from its center and radius. If the argument 4 is 0, the arguments 5 and 6 are
taken into account.
Otherwise, a circle is created.
Example
Geometrical Set.1\Circle.1 =
circleCtrRadius(Geometrical Set.1\Point.1 , `zx plane` ,20mm,0,10deg,320deg)

circleCtrPt(*center*: Point, *point*: Point, *support*: Surface, *radius*: Length, *limits*: Integer, *start*: Angle, *end*: Angle): Circle

Creates a circular arc from its center and another point located on the circle. If the argument 4 is 0, the arguments 5 and 6 are taken into account. Otherwise, a circle is created.
Example
Geometrical Set.1\Circle.2 =
circleCtrPt(Geometrical Set.1\Point.1 , Geometrical Set.1\Point.2 ,
`xy plane` ,1,10deg, 370deg)

circle2PtsRadius(*point1*: Point, *point2*: Point, *support*: Surface, *radius*: Length, *orientation*: Boolean, *limits*: Integer): Circle

Creates a circular arc. The points specified in the arguments 1 and 2 are located on the arc to be created and define the arc limits when the integer specified in the argument 6 is 0. When 0 is specified in the argument 6, modifying the argument 5 boolean value allows you to display the alternative arc.
Example
Geometrical Set.1\Circle.3 =
circle2PtsRadius(Geometrical Set.1\Point.1 ,Geometrical Set.1\Point.2 ,
`xy plane`,50mm, true, 0)

Circle3Pts (pt1: Point, pt2: Point, pt3: Point, Limits: Integer) : Circle

Creates one or more circular arcs passing through three points. When 0 is specified in the argument 4, the first and third points define the arc limits. When 1 is specified in the argument 4 the whole circle is defined. When 2 is specifies in the argument 4 the direct circle is defined. When 3 is specified in the argument 4, the complementary circle is defined.
Example
Geometrical Set.1\Circle.2 =
circle3Pts(Geometrical Set.1\Point.1, Geometrical Set.1\Point.2,
Geometrical Set.1\Point.3, 0)

circleBitgtRadius(*crv1*:Curve, *crv2*:Curve, *support*: Surface, *radius*: Length, *orientation1*: Boolean, *orientation2*: Boolean, *Limits*: Integer) : Circle

Creates one or more circular arcs tangent to two curves. When 0 is specified in the argument 7, the tangency points define the arc limits. Modifying the *orientation1* argument value allows you to reverse the arc orientation with respect to the *crv1* curve (there may be no solution). Modifying the *orientation2* argument value allows you to reverse the arc orientation with respect to the *crv2* curve.

Example

```
Geometrical Set.1\Circle.4 =
circleBitgtRadius(Geometrical Set.1\Circle.2 ,Geometrical Set.1\Circle.5 ,`xy plane`,
 30mm, false, false, 0)
```

circleBitgtPoint(crv1:Curve, crv2:Curve, pt:Point , support: Surface, orientation1: Boolean, orientation2: Boolean, Limits: Integer) : Circle

Creates one or more circular arcs tangent to two curves and passing through a point on the second curve. When 0 is specified in the argument 7, the tangency points define the arc limits. Modifying the orientation1 argument value allows you to reverse the arc orientation with respect to the crv1 curve (there may be no solution). Modifying the orientation2 argument value allows you to reverse the arc orientation with respect to the crv2 curve.

Example

```
Geometrical Set.1\Circle.4 =
circleBitgtPoint(Geometrical Set.1\Circle.2 ,Geometrical Set.1\Circle.5,Geometrical Set.1\Point.1 ,
`xy plane`, false, false, 0)
```

circleBitgtradius(curve: Curve, point: Point, support; Surface, radius: Length, orientation1: Boolean, orientation2: Boolean, limits: Integer : Circle

Creates one or more circular arcs tangent to two curves.

circleTritgt(*crv1*:Curve, *crv2*:Curve, *crv3*:Curve, *support*: Surface, *radius*: Length, *orientation1*: Boolean, *orientation2*: Boolean, *orientation3*: Boolean, *Limits*: Integer) : Circle

Creates one or more circular arcs tangent to three curves. When 0 is specified in the argument 9, the tangency points define the arc limits. Modifying the value of an *orientation* argument allows you to reverse the arc orientation with respect to the curve which has the same order in the argument specification (*orientation1* to be associated with *crv1*).

Example

```
Geometrical Set.1\Circle.6 =
circleTritgt(Geometrical Set.1\Circle.2 ,Geometrical Set.1\Circle.7 ,Geometrical Set.1
\Circle.5 , `xy plane` ,false,false,false,1)
```

List

List methods are used to manage lists of parameters, pads ...: They enable the user to create lists, to add items to the list, to remove items from the list, to retrieve values from the list, to move elements of the list to another position, to filter, and to copy the content of a list into another one.

- **List.Size** () **:** Integer
  Method used to return the number of items contained in the list.

- **List.AddItem** (*Object: Objecttype*, *Index: Integer*):VoidType
  Method used to add an item to the list.

```
let list (List)
list.AddItem(PartBody\Hole.2 ,1)
list.AddItem(PartBody\Hole.3 ,2)
Message("#",list.Size())
```

- **List.RemoveItem** (*Index: Integer*) :VoidType
  Method used to remove an item from the list.

- **List.GetItem** (*Index: Integer*) :ObjectType
  Method used to retrieve a value/item from the list

- **List.ReorderItem** (*Current: Integer, Target: Integer*) :ObjectType
  Method used to move an element of the list to a new position.

- **Copy** (List: List) : List
  Method used to copy the content of a list and paste it in another list.

- **List** (Next: ObjectType, ...): List
  Method used to create a list.

- **List.Sum** (): Real
  Computes the sum of the items contained in the list..

- **List.IndexOf** (Element: ObjectType, StartIndex:Integer):Integer
  Returns the index of a list item.

- **Compute**()
  Function used to compute the result of an operation performed on the attributes supported by the features contained in the list.
  Example: List.1 .Compute("+","Hole","x.Diameter",Length.1)
  Where:
- List.1 is the name of the list on which the calculation will be performed
- + is the operator used. (Supported operators are: -, min, and max.)
- Hole is the type of the list items used for the calculation (to calculate the diameter, the type to be indicated is Hole, to calculate the volume, the type to be indicated is Solid)
- x stands for the list items. Note that the type of the items contained in the list should be identical.
- Length.1 is the output parameter.

- **Filter()**
  Method used to filter a list of objects by extracting the objects that fulfill a Boolean expression.
  This method has the following signature:
  List->Filter(String TypeName,String Expression) : List
    - TypeName is the type of the objects that the user wants to extract (in can be "". In this case, no filtering is done on types)
    - The second string Expression corresponds to the Boolean expression that must fulfill the objects of this given type. In this expression "x" is used as the variable name of type TypeName. This string can be equal to "". In this case, no expression is checked.

Example:
```
I = (List->Filter("BNS","x.Diameter > 3mm"))->Size()
I = (List->Filter("BNS",""))->Size()
```

Measures

Measures are functions that compute a result from data captured from the geometry area. Measures are application-related objects and they won't be displayed in the dictionary if you don't have the right product installed (Part Design or Generative Shape Design for example).

Sample: KwrMeasuresWiz.CATPart

- **distance***(Body1, Body2)* **:** Length
  Returns the distance between two bodies of a part.

- **minimumCurvatureRadius***(Curve):Length*
  For an item of dimension 1 (a curve), enables the user to measure its minimum radius of curvature.

- **nbDomains***(Body)***: Integer**
  For all types of items, enables the user to compute the number of domains.

- **length**(*GSMCurve*) :**Length**
  Returns the total length of a curve.

- **length**(*GSMCurve*, *Point1*, *Point2*) : **Length**
  Returns the length of a curve segment delimited by *Point1* and *Point2*.

- **length**(*GSMCurve*,*Point1*,*Boolean*): **Length**
  Returns the length of a curve segment located between *Point1* and one of the curve ends.
  Modifying the boolean value allows you to retrieve the length from the specified point to the other end.

- **area**(*Surface*): Area
  Returns the area of a surface generated by the Generative Shape Design product (an extruded surface for example).

- **area**(*Curve*) : Area
  Returns the area delimited by a curve.

- **perimeter**(*Surface*,...):Length
  Returns the perimeter of a surface. It can take several surface features in input. The perimeter function sums up the perimeter of each surface. The returned value is a length.

- ***point*.coord**(*Integer*): Length
  Returns the coordinate of a point. Returns X if 1 is specified, Y if 2 is specified, Z if 3 is specified.

- ***point*.coord**(*oX*: Length, *oY*: Length, *oZ:* length): Void
  Assigns the point coordinates to the length parameters specified in the arguments. This method can only be used in Knowledge Advisor rules.

- **Body.centerofgravity**(x: out length, x: out length, z: out length): Void Type
  Enables the user to compute the center of gravity.

- **volume**(*closedSurface*) : Volume
  Returns the volume of a closed surface.

- **angle**(*Center: Point1, Pt1: Point2, Pt2*) : Angle
  Returns the angle between the lines "C-Point1" and "C-Point2".

- **angle**(*Direction*,*Direction*) : Angle
  Returns the angle between two directions.

- **angle**(*Line*, *Line*) : Angle
  Returns the angle between the *Line1* and *Line2* lines.

- **angle**(*Plane*, *Plane*) : Angle
  Returns the angle between t2 planes.

- **angleoriented***(Direction, Direction, Direction)***: Angle**
  Returns the angle between 2 directions and oriented by a third direction.

- **angleoriented**(*Line, Line, Direction*)**: Angle**
  Returns an angle between 2 lines and oriented by the direction.

- **angleoriented**(*Plane, Plane, Direction*)**: Angle**
  Returns an angle between 2 planes and oriented by the direction.

- **curvature**(*crv*: Curve, *pt*: Point): Real
  Returns the curvature of a curve in a given point.

- **distancedir** (*Body, Body, Direction*): Length
  Returns the distance between two bodies of a part and oriented by the direction

Surface Constructors

Offset	assemble	split (surface, surface, boolean)
split (surface, curve, boolean)	trim(*surface*, *boolean*, *surface*, *boolean*)	near(surface, wireframe) : Surface
extrude(*curve*, *direction*, *length*, *length*, *boolean*) : Surface	extrude(*surface*, *direction*, *length*, *length*, *boolean*) : Surface	revolve(*curve*, *line*, *angle*, *angle*) : Surface
revolve(*surface*, *line*, *angle*, *angle*) : Surface	loft(*sections*: list, *orientations*: list)	loft(*sections:* list, *orientations*: list, *guides*: list)

- **offset**(*surface*, *length*, *boolean*) : Surface
  Creates an offset surface.Set orientation boolean to false to change the side of the created surface regarding the reference surface.

Example
Geometrical Set.1\Surface.2=
offset(Geometrical Set.1\Sweep.1, 10mm, false)

- **assemble**(*surface*, ...) : Surface
  Creates a join of several surfaces.

Example
Geometrical Set.1\Surface.2=
assemble(Geometrical Set.1\Sweep.1, Geometrical Set.1\Sweep.2, Geometrical Set.1\Offset.2)

- **split**(*surface*, *surface*, *boolean*) : Surface
  Creates a split of one surface by another. Use the third argument to choose the side to keep.

Example
Geometrical Set.1\Surface.2=
split(Geometrical Set.1\Sweep.1, Geometrical Set.1\Sweep.2, true)

- **split**(*surface*, *curve*, *boolean*) : Surface
  Creates a split of one surface by a curve. Use the third argument to choose the side to keep.

Example
Geometrical Set.1\Surface.2=
split(Geometrical Set.1\Sweep.1, Geometrical Set.1\Curve.2, true)

- **trim**(*surface*, *boolean*, *surface*, *boolean*) : Surface
  Creates a trim of one surface by another. Use the Booleans to choose the side to keep on each surface.

Example
Geometrical Set.1\Surface.2=
trim(Geometrical Set.1\Sweep.1, false, Geometrical Set.1\Sweep.2, true)

- **near**(surface, wireframe) : Surface
  Extracts a connex sub element of a non connex entity which is the nearest from another element.

Example
Geometrical Set.1\Surface.2=
near(Geometrical Set.1\Sweep.1, point(0mm,50mm,0))

- **extrude**(*curve*, *direction*, *length*, *length*, *boolean*) : Surface
  Extrudes a wireframe profile in a given direction.

Example
Geometrical Set.1\Surface.2=
extrude(Geometrical Set.1\Sketch.1, direction(1,0,0), 0mm, 50mm, true)

- **extrude**(*surface*, *direction*, *length*, *length*, *boolean*) : Surface
  Extrudes a surface in a given direction. The result is the skin of the generated volume.

Example
Geometrical Set.1\Surface.2=
extrude(Geometrical Set.1\Surface.1, direction(1,0,0), 0mm, 50mm, true)

- **revolve**(*curve*, *line*, *angle*, *angle*) : Surface
  Revolves a wireframe profile around a given axis.

Example
Geometrical Set.1\Surface.2=
revolve(Geometrical Set.1\Sketch.1, Geometrical Set.1\Line.1, 0deg, 90deg)

- **revolve**(*surface*, *line*, *angle*, *angle*) : Surface
  Revolves a surface around a given axis. The result is the skin of the generated volume.

Example
Geometrical Set.1\Surface.2=
revolve(Geometrical Set.1\Surface.1, Geometrical Set.1\Line.1, 0deg, 90deg)

- **loft**(*sections*: list, *orientations*: list)
  Creates a loft from several sections.

Example
Geometrical Set.1\Surface.2=
loft(List(Geometrical Set.1\Sketch.1,Geometrical Set.1\Sketch.2), List(1,1))

- **loft**(*sections:* list, *orientations*: list, *guides*: list)
  Creates a loft from several sections and several guides.

Example

Geometrical Set.1\Surface.2=
loft(List(Geometrical Set.1\Sketch.1,Geometrical Set.1\Sketch.2), List(1,1), List(Geometrical Set.1\Line.1,Geometrical Set.1\Line.2))

Wireframe Constructors

spline	intersect	intersect	intersect
curveparallel	project	project	project
assemble	corner	split	trim
near	near	extrude	revolve

- **spline**(pt: Point, ...): Curve
  Creates a spline from several points.

Example
Geometrical Set.1\Curve.1 =
spline(Geometrical Set.1\Point.1, Geometrical Set.1\Point.2, Geometrical Set.1\Point.3)

- **intersect**(*crv*: Curve, *crv*: Curve) : Point
  Constructs the point where two curves intersect.

Example
Geometrical Set.1\Point.6 =
intersect(Geometrical Set.1\Curve.1 ,Geometrical Set.1\Curve.2)

- **intersect**(*crv*: Curve, *su*: Surface) : Point
  Constructs the point where a curve and a surface intersect.

Example
Geometrical Set.1\Point.7 =
intersect(Geometrical Set.1\Spline.1 ,Geometrical Set.1\Extrude.1)

- **intersect**(*su*: Surface, *su*: Surface) : Curve
  Constructs the curve where two surfaces intersect.

Example
Geometrical Set.1\Curve.4 =
intersect(Geometrical Set.1\Extrude.2 ,Geometrical Set.1\Extrude.1)

- **curveparallel**(*crv*: Curve, *su*: Surface, offset: Length) : Curve
  Constructs the curve parallel to another curve. The surface specified in the second argument is the support.

Example
Geometrical Set.1\Curve.4 =
curveparallel(Geometrical Set.1\Spline.1 , Geometrical Set.1\Extrude.2 ,20mm)

- **project**(toproject: Point, support: Curve): Point
  Projects a point on a curve.

Example:
Geometrical Set.1\Point.3=
project(`Geometrical Set.1\Point.2` ,`Geometrical Set.1\Sketch.2`)

- **project**(toproject: Point, support: Surface): Point
  Projects a point on a surface.

- **project**(toproject: Point, support: Surface): Surface
  Projects a curve on a surface.

- **assemble**(Curve,...):Curve
  Creates a join of several curves.

- **corner**(*crv1*: Curve, *crv2*: Curve, *support*: Surface, radius: Length, orientationcrv1: Boolean, orientationcrv2: Boolean, trim: Boolean : Curve
  Constructs a corner between two curves. The arguments 5 and 6 should be used to scan the possible solutions. See the *Generative Shape Design User's Guide* for more information on corners.

Example
Geometrical Set.1\Curve.6 =
corner(Geometrical Set.1\Curve.1 ,Geometrical Set.1\Curve.2, `xy plane` , 50mm,true,true,false)

- **split**(tosplit:curve, splitting: Wireframe, orientation:Boolean):Curve
  Enables to split a surface.

Example
Geometrical Set.1\Curve.2=
split(`Geometrical Set.1\Sketch.2` ,`Geometrical Set.1\Point.3` ,TRUE)

- **trim**(crv1: Curve, orientationCrv1: Boolean, orientationCrv1: Boolean, crv1: Curve, orientationCrv2, Boolean): Curve
  Enables to trim two two wireframe elements.

Example
Geometrical Set.1\Curve.1=
trim(`Geometrical Set.1\Sketch.3` ,TRUE,`Geometrical Set.1\Sketch.2` ,FALSE)

- **near**(crv:Curve, near:Wireframe): Curve
  Creates the nearest entity of several sub-element. The result is a curve.

- **near**(crv:Point, near:Wireframe): Point
  Creates the nearest entity of several sub-element. The result is a point.

- **extrude**(Point, Direction, length1: Length, length2: Length, orientation: Boolean): Line
  Creates a line. Extrusion of a point depending on a direction.

- **revolve**(Point, axis, Line: angle1, Angle: angle2, Angle): Circle
  Enables to create a circle by revolving a point according to a given direction.

Example:

```
Geometrical Set.1\Curve.3=
revolve(`Geometrical Set.1\Point.3` ,`Geometrical Set.1\Extrude.1\Direction.2` ,10deg,20deg)
```

Part Measures

smartVolume and smartWetarea refer to intermediate states of a solid. smartVolume does not compute the volume of each pad contained in a PartBody but the total volume.
Example: Given a PartBody containing 3 pads: The volume of Pad.1 = 0.1m3, The volume of Pad.2=0.1m3 and the volume of Pad.3=0.1m3. The Volume of Pad.3 displayed will be Pad.3=0.3M3. The volume of Pad.3=the Volume of Pad.1+ the volume of Pad.2+ the volume of Pad.3.

Note that this applies also to smartWetarea (the total wet area is computed).

- **smartVolume***(elem: Solid, ...)*: Volume
  Returns the volume of a solid.

  Example
  Total_Volume=
  smartVolume(PartBody)

- **smartWetarea***(elem: Solid, ...)* : Area
  Returns the wet area of a solid.

  Example
  Total_Area=
  smartWetarea(PartBody\Pad.1)

Plane Constructors

- **plane**(*point*, *point*, *point*) : Plane
  Creates a plane through 3 points.

Example
Geometrical Set.1\Plane.1=
plane(Geometrical Set.1\Point.1,Geometrical Set.1\Point.2,Geometrical Set.1\Point.3)

- **plane**(*a*:Real, *b*:Real, *c*:Real, *d*:Length) : Plane
  Creates a plane from its equation aX+bY+cZ=d.

Example
Geometrical Set.1\Plane.1=
plane(1,0,0,50mm)
creates the plane of X=50mm equation.

- **plane**(*line*, *line*) : Plane
  Creates a plane through 2 lines.

Example
Geometrical Set.1\Plane.1=
plane(Geometrical Set.1\Line.1,Geometrical Set.1\Line.2)

- **plane**(*point*, *line*) : Plane
  Creates a plane through a point and a line.

Example
Geometrical Set.1\Plane.1=
plane(Geometrical Set.1\Point.1,Geometrical Set.1\Line.1)

- **plane**(*curve*) : Plane
  Creates a plane through a planar curve.

Example
Geometrical Set.1\Plane.1=
plane(Geometrical Set.1\Curve.1)

- **planetangent**(*surface*, *point*) : Plane
  Creates a plane tangent to a surface at a point.

Example
Geometrical Set.1\Plane.1=
planetangent(Geometrical Set.1\Sweep.1, Geometrical Set.1\Point.1)

- **planenormal**(*curve*, *point*) : Plane
  Creates a plane normal to a curve at a point.

Example
Geometrical Set.1\Plane.1=
planenormal(Geometrical Set.1\Spline.1, Geometrical Set.1\Point.1)

- **planeoffset**(*plane*, *length*, *boolean*) : Plane
  Creates an offset plane from another at a given distance. Set orientation boolean to false to change the side of the created plane regarding the reference plane.

Example
Geometrical Set.1\Plane.2=
planeoffset(Geometrical Set.1\Plane.1, 50mm, false)

- **planeoffset**(*plane*, *point*) : Plane
  Creates an offset plane from another passing through a point.

Example
Geometrical Set.1\Plane.2=
planeoffset(Geometrical Set.1\Plane.1, Geometrical Set.1\Point.1)

- **planeangle**(*plane*, *line*, *angle*, *boolean*) : Plane
  Creates an angle plane. Set orientation boolean to false to change the side of the created plane regarding the reference plane.

Example
Geometrical Set.1\Plane.2=
planeangle(Geometrical Set.1\Plane.1, Geometrical Set.1\Line.1, 30deg, true)

- **planemean**(Point,...): Point
  Computes a mean plane from a set of points.

Analysis operators

- **energy** (*Case: StaticSolution*)
  Computes the global energy in a static case solution.

- **misesmax** (*Case: StaticSolution*)
  Computes the maximum value of the nodal VonMises stress.
  Example
  misesmax.1=misesmax("Finite Element Model\Static Case Solution.1")

- **dispmax** (*Case: StaticSolution*)
  Computes the nodal maximum displacement.
  Example
  length.1=dispmax("Finite Element Model\Static Case Solution.1")

- **frequency** (*Case: FrequencySolution*)
  Computes a given frequency.
  Example
  Frequency.1=Frequency("Finite Element Model\Frequency Case Solution.1")

- **frequencies** (*Case: FrequenciesSolution*)
  Computes all the frequencies.
  Example
  FrequenciesList.1=Frequencies("Finite Element Model\Frequencies Case Solution.1")

- **globalerror** (*Case: StaticSolution*)
  Computes the global error percentage of a static case.
  Example
  percentage.1=globalerror("Finite Element Model\Static Case Solution.1")

- **bucklingfactors** (*Case: BucklingSolution*)
  Computes a list of buckling factors.
  Example
  Bucklingfactors.1=BucklingFactors("Finite Element Model\Buckling Case Solution.1")

- **dispmaxongroup** (*Case: AnalysisResults, Group:Group*): Length
  Computes the nodal maximum displacement. It applies to a group of items.

Mathematical Functions

Sample (illustrates interpolations): KwrInterpolations.CATPart

- **abs**(Real): Real
  Calculates the absolute value of a number.
- **ceil**(Real): Real
  Returns the smallest integer value that is greater than or equal to the value specified in the argument.
- **floor**(Real):Real
  Returns the largest integer value that is less than or equal to the value specified in the argument.
- **int**(Real):Real
  Returns the integer value of a number.
- **let**
  Assigns a value to a temporary variable (let x = 30 mm)
- **min**(Real,Real):Real, **max**(Real,Real)
  Returns the minimum or maximum of a set of values specified in the argument.
- **sqrt**(Real):Real
  Returns the square root.
- **log**(Real):Real
  Returns the logarithm.
- **ln**(Real):Real
  Returns the natural logarithm.
- **round**(Real):Real
  Returns a rounded number.
- **round**(Real, String, Integer):Real
  Returns a rounded number.
  - For Real = 13.552mm
  - String = m (for meter)
  - Integer = 2

  The returned number is 13 mm
- **exp**(Real):Real
  Returns the exponential.
- **LinearInterpolation**(*arg1*:Real, *arg2*:Real, *arg3*:Real) : Real
  Should be used when creating a parallel curve from a law.
  Example:
  1 - Create a line in the Generative Shape Design workbench
  2 - Access the Knowledge Advisor workbench and create the law below:
  FormalReal.1 = LinearInterpolation(1,9,FormalReal.2)
  3 - Back to the Generative Shape Design, create a parallel curve. Select the Law mode and specify the law above as the one to be applied.

- **CubicInterpolation**(*arg1*:Real, *arg2*:Real, *arg3*:Real) : Real
  Should be used when creating a parallel curve from a law.
  Example:
  1 - Create a line in the Generative Shape Design workbench
  2 - Access the Knowledge Advisor workbench and create the law below:
  FormalReal.1 = CubicInterpolation(1,50,FormalReal.2)
  3 - Back to the Generative Shape Design, create a parallel curve. Select the Law mode and specify the law above as the one to be applied.

- **mod(Real,Integer): Real**
  Enables the user to retrieve the remainder of the division of the integer part of the real number by the integer.

- **Cos**(Real):Real**, cosh** (Real): Real
  Calculates the cosine(cos) or hyperbolic cosine(cosh).
  Example
  Real.1 = cos(PI*1rad/4)
  Real.1 = cos(45deg)

- **tan**(Real): Real, **tanh**(Real): Real
  Calculates the tangent(tan) or hyperbolic tangent (tanh).

- **sin**(Real):Real, **sinh**(Real):Real
  Calculates the sine or hyperbolic sine.

- **asin**(Real):Real, **asinh**(Real):Real
  Calculates the arcsine or hyperbolic arcsine.

- **acos**(Real):Real, **acosh**(Real):Real
  Calculates the arccosine or hyperbolic arccosine.

- **atan**(Real):Real, **atanh**(Real):Real
  Calculates the arctangent or hyperbolic arctangent.

For these methods to be efficient, you should use real numbers only.

Electrical User Functions in Knowledge Products

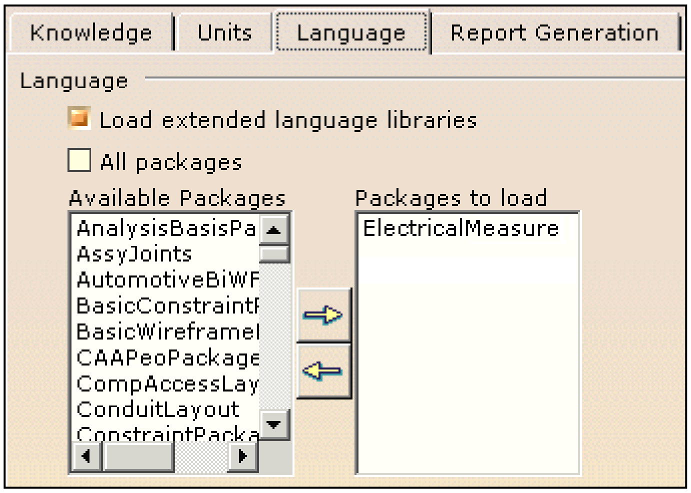
About the Electrical User Functions...

- Elec_DistanceCommon
- DistanceWireProduct
- VisualMode

To be able to use these functions, you need to activate the **ElectricalMeasure** package.

To do so:

1. Select **Tools -> Options... -> General -> Parameters and Measures** and go to the **Language** tab.
2. Choose the **ElectricalMeasure** package and click the right arrow:



3. Click **OK** to validate.

Elec_DistanceCommon

Syntax

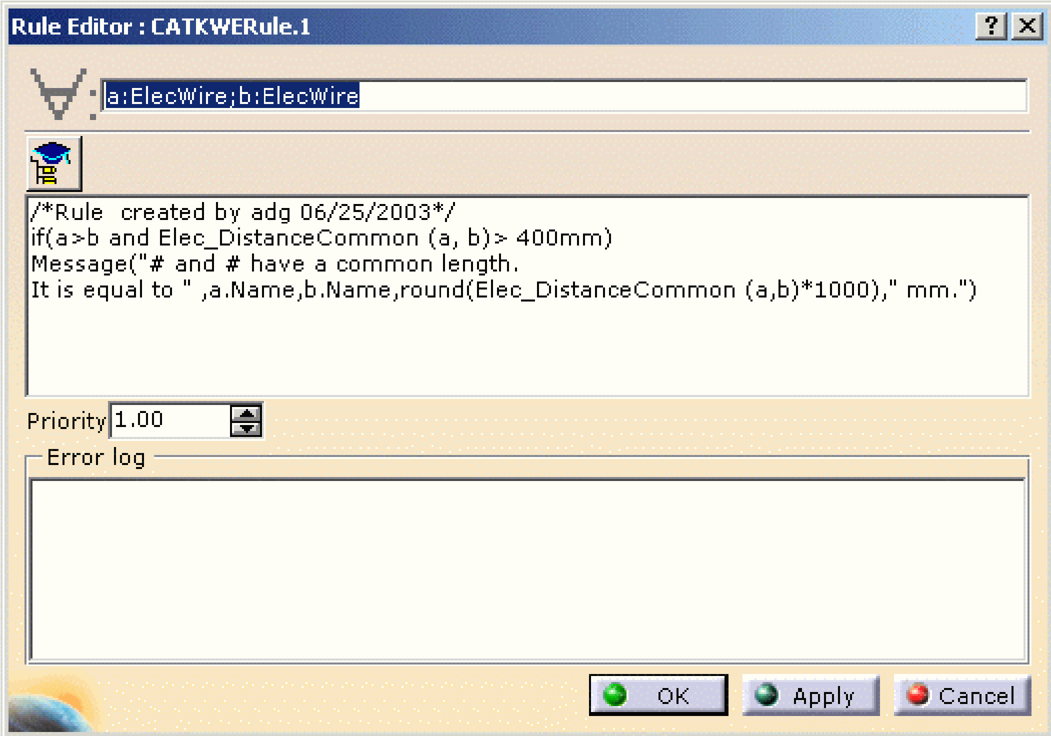
Elec_DistanceCommon(*Wire1: Wire, Wire2: Wire*):**Length**

Returns the common length of the two wires given as input arguments.

The type of **Wire1** and **Wire2** is ElecWire.

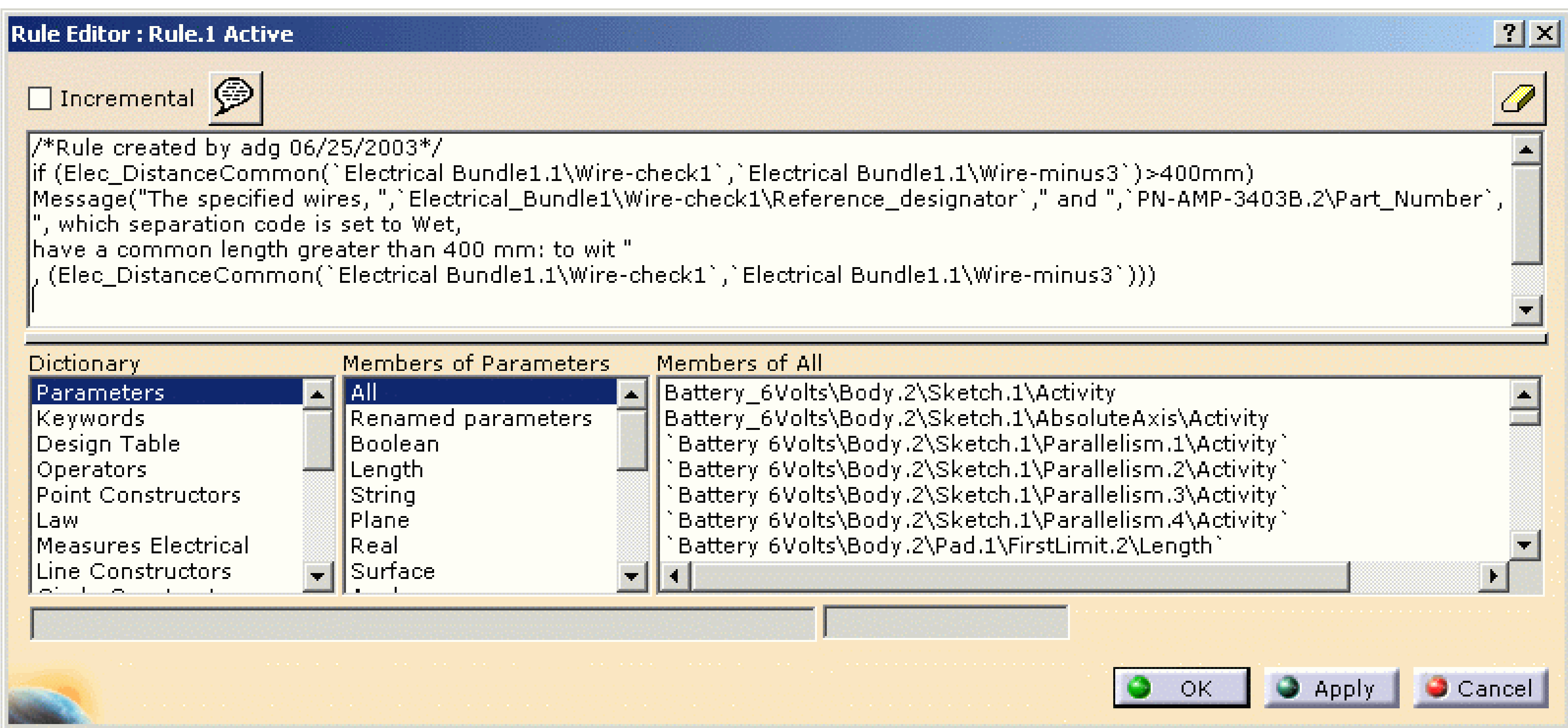
Example 1

The **Elec_DistanceCommon** user function can be used in Knowledge Expert to find all the couples of wires in the session that have a common length greater than a given value.

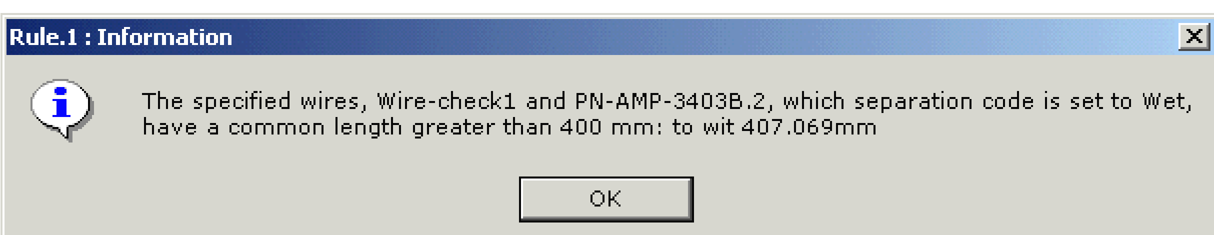


Example 2

In Knowledge Advisor, it can be used to define a rule giving the common length of two specific wires sharing properties.

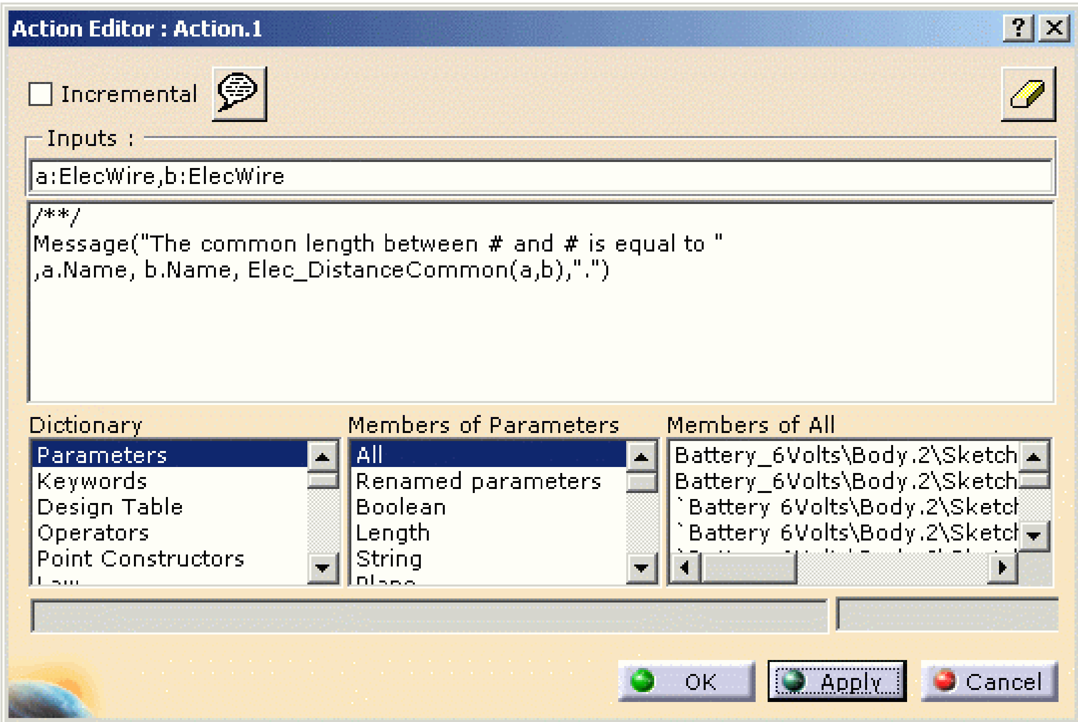


Applying the rule displays the following message if the condition is met:

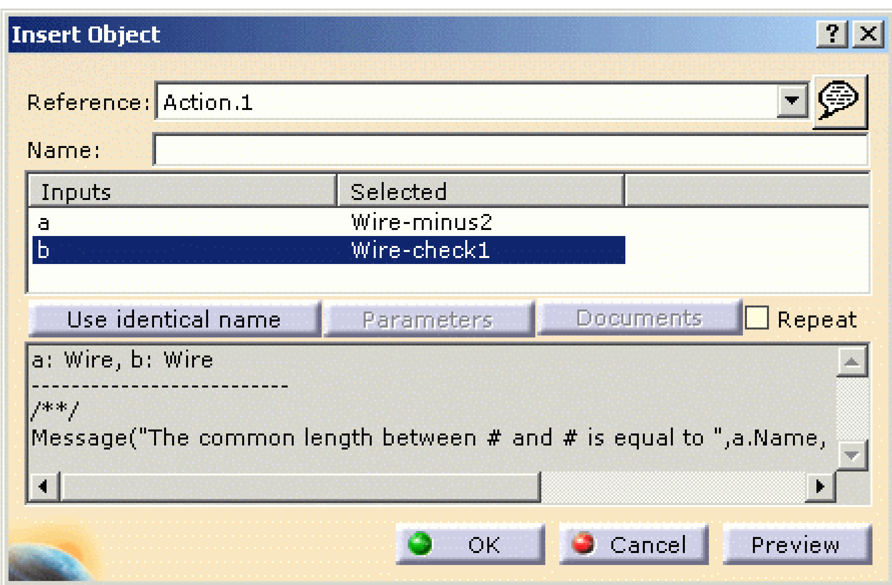


Example 3

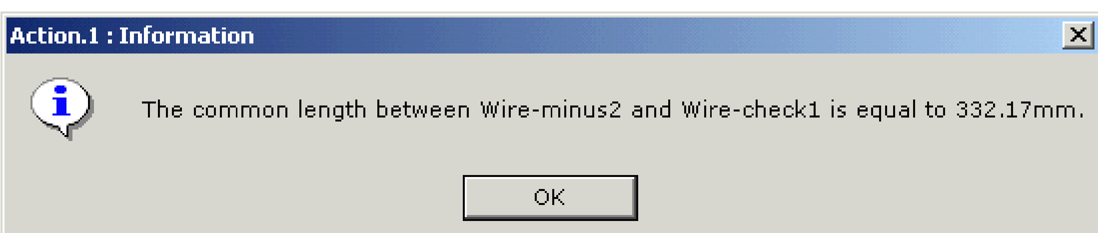
Still in Knowledge Advisor, to verify that two wires selected in the specification tree have a common length, the following action can be defined:



then ran: select two wires in the specification tree and click **OK** to validate.



The following message displays:



DistanceWireProduct

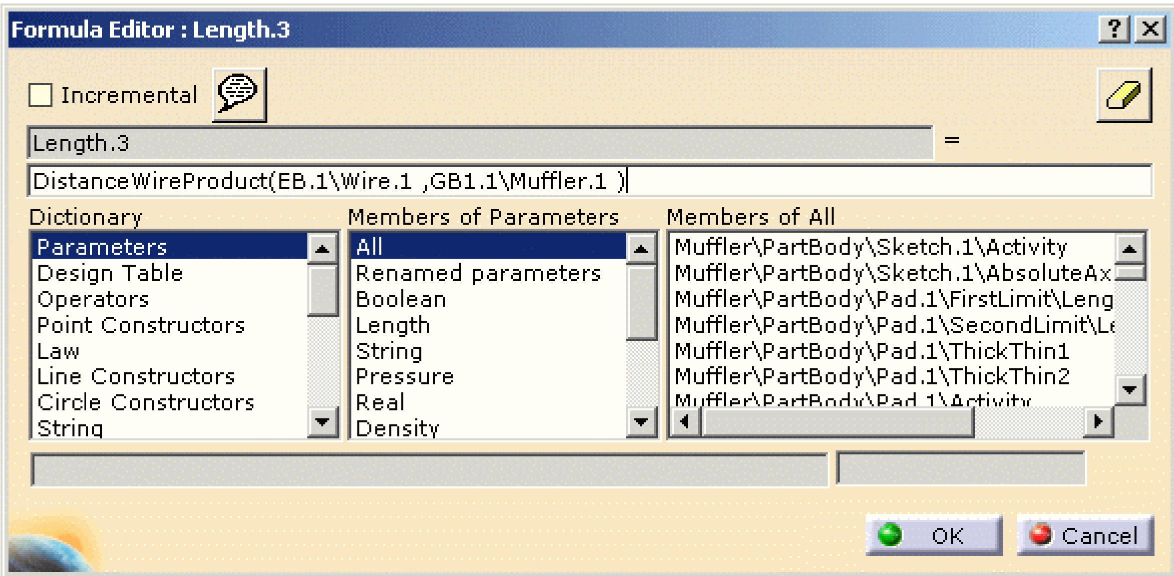
Syntax

DistanceWireProduct(*Wire1: Wire, Object: Product*):**Length**

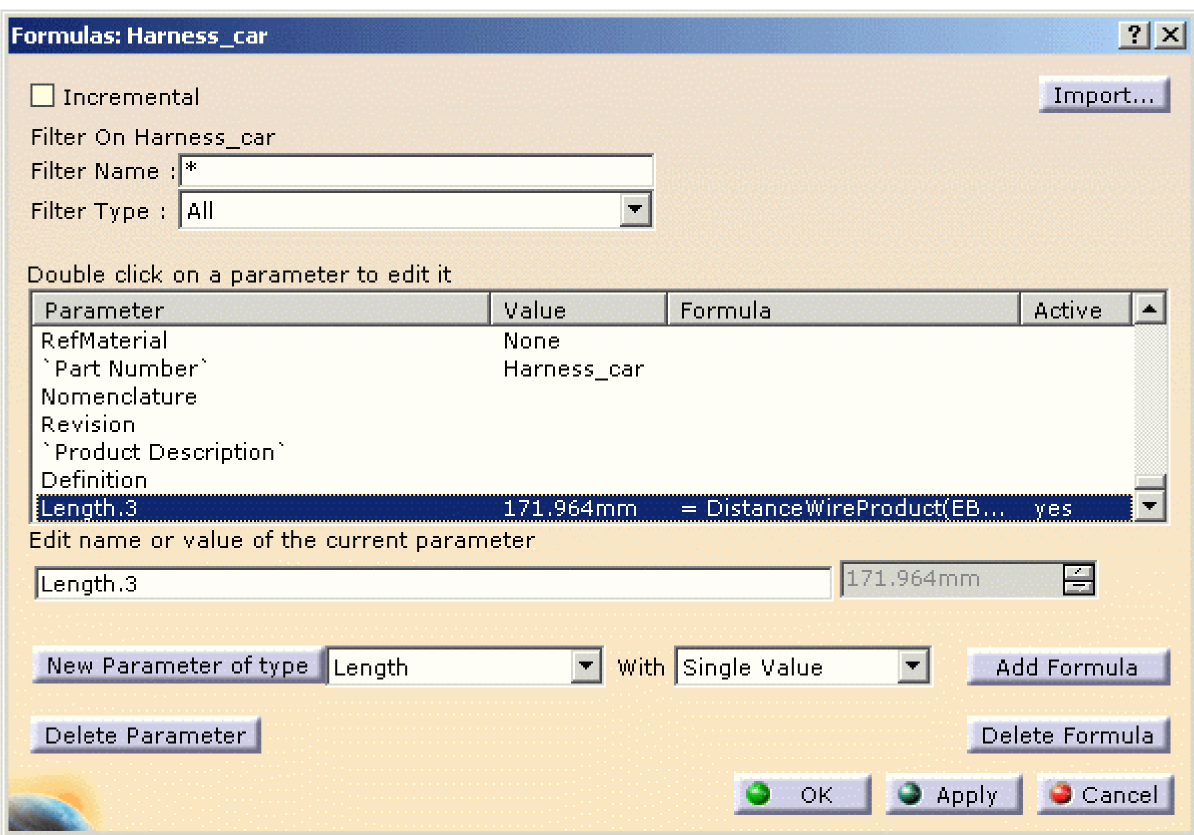
Returns the minimum length between a wire and a product in session. The product must contain at least one part.

Example 1

The **DistanceWireProduct** user function can be used with the f(x) command to display the distance between a wire and a component in the session.



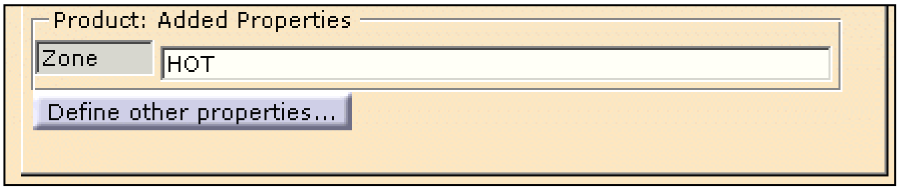
This formula returns the following value:



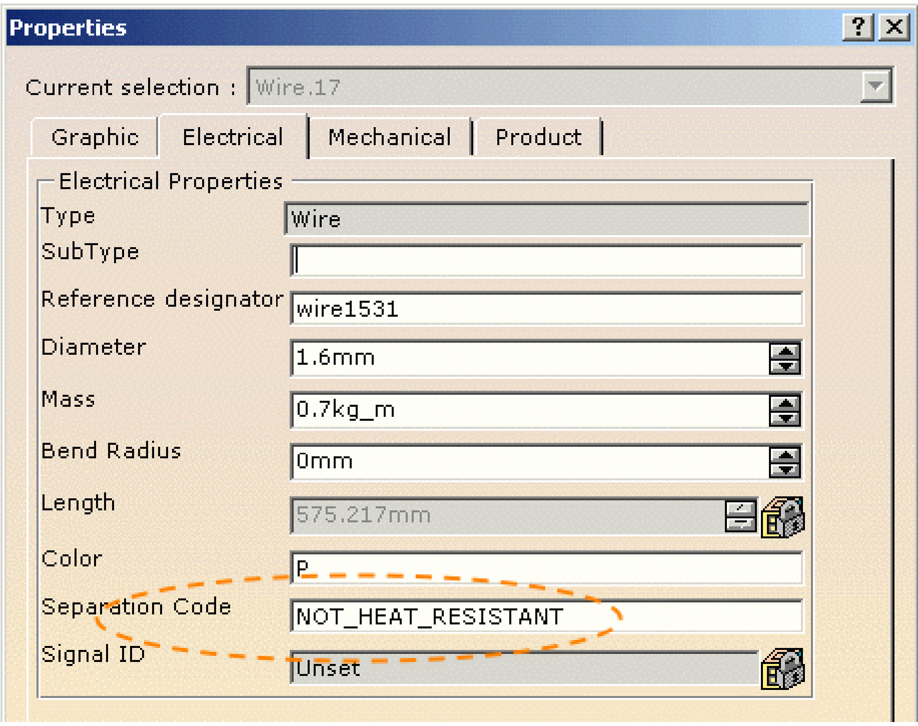
Example 2

The **DistanceWireProduct** user function can be used in Knowledge Expert to find all the wires in the session that have a minimum distance to defined components smaller than a critical value chosen by the user. The components can be defined as heat-resistant.

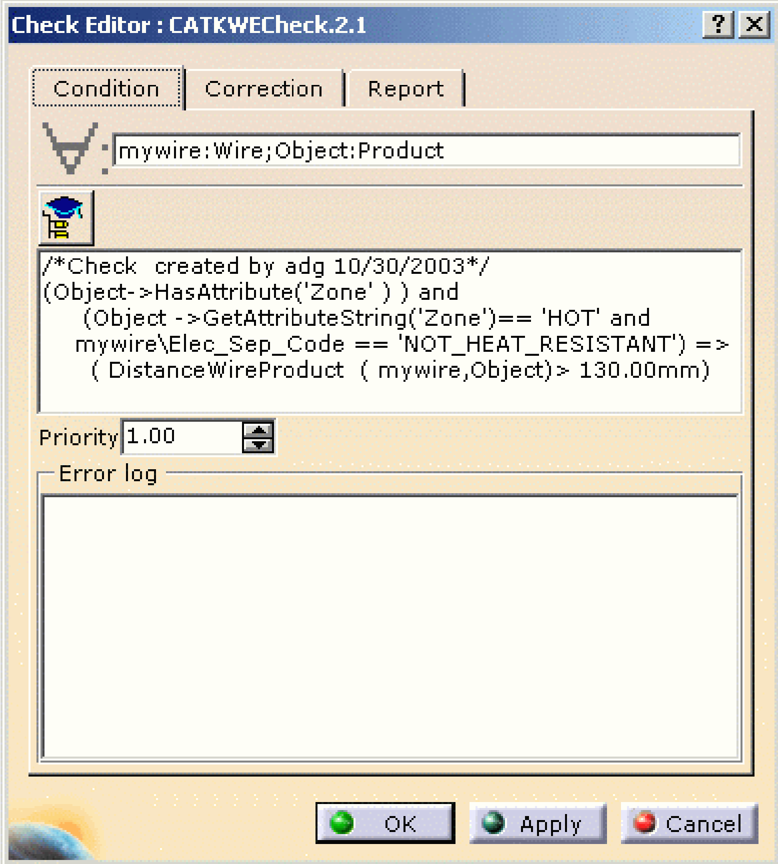
Properties have been added to the product:



and to the wires:



A check is defined as follows:



Updating the session displays green/red light on the check: CATKWECheck.2.1

A report is generated showing the check result: some wires verify the condition, other not.

Current Document Name

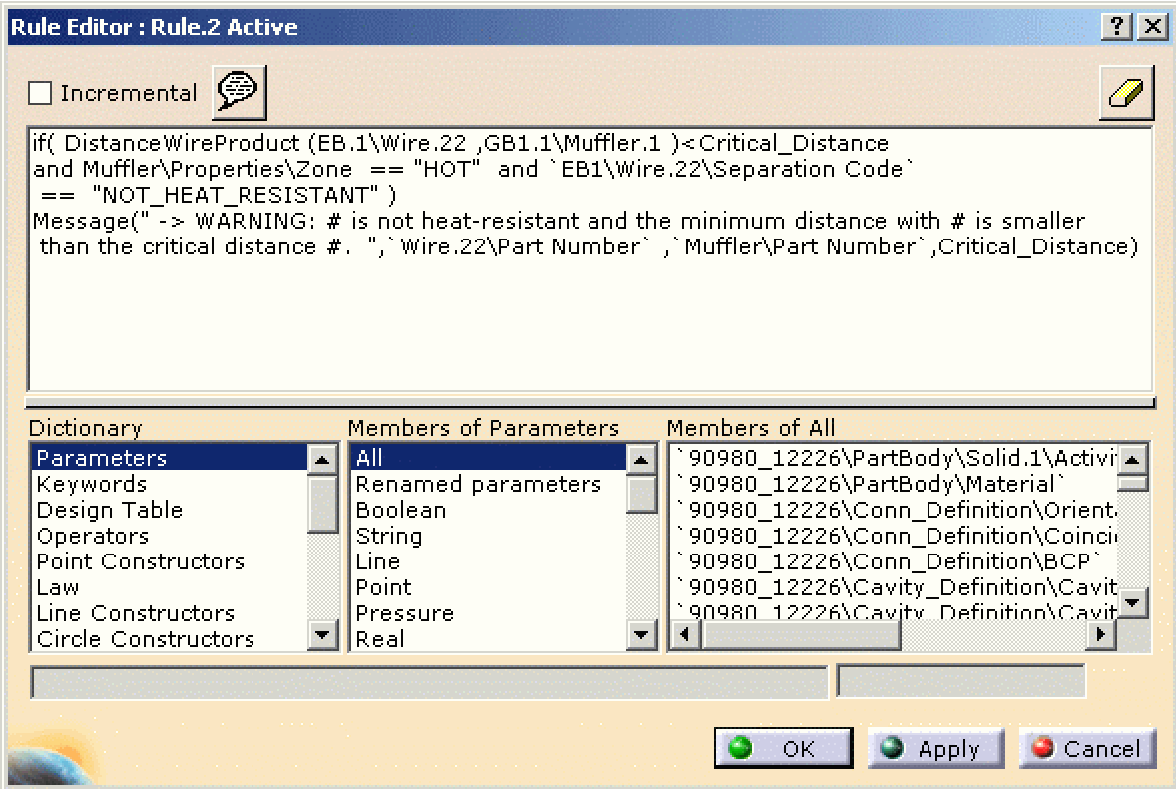
E:\users\adg\Electrical\V5R13\Knowledge\Main.CATProduct 10/31/2003

✗	RuleBase\CATKWERuleSet.1	CATKWECheck.2.1	55%		

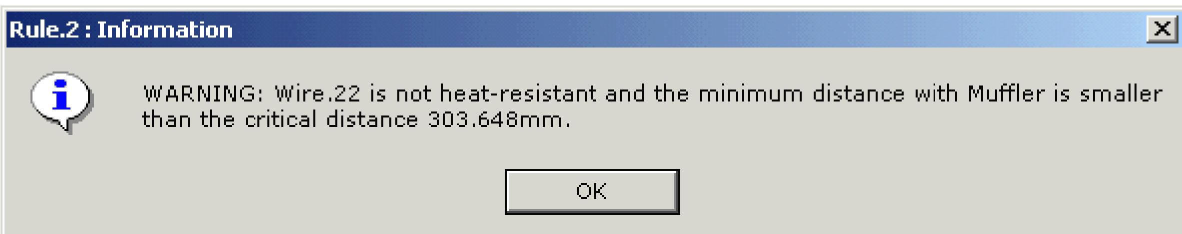
✗	Harness_car\EB.1\Wire.17	Harness_car\GB1.1\Muffler.1
✗	Harness_car\EB.1\Wire.13	Harness_car\GB1.1\Muffler.1
✗	Harness_car\EB.1\Wire.10	Harness_car\GB1.1\Muffler.1
✗	Harness_car\EB.1\Wire.6	Harness_car\GB1.1\Muffler.1
✓	Harness_car\EB.1\Wire.22	Harness_car\GB1.1\Muffler.1
✓	Harness_car\EB.1\Wire.20	Harness_car\GB1.1\Muffler.1
✓	Harness_car\EB.1\Wire.19	Harness_car\GB1.1\Muffler.1
✓	Harness_car\EB.1\Wire.18	Harness_car\GB1.1\Muffler.1
✓	Harness_car\EB.1\Wire.2	Harness_car\GB1.1\Muffler.1

Example 3

The **DistanceWireProduct** user function can be used in Knowledge Advisor to define a rule that displays a warning message if a minimum distance between a wire and an object is smaller than a critical value chosen by the user.

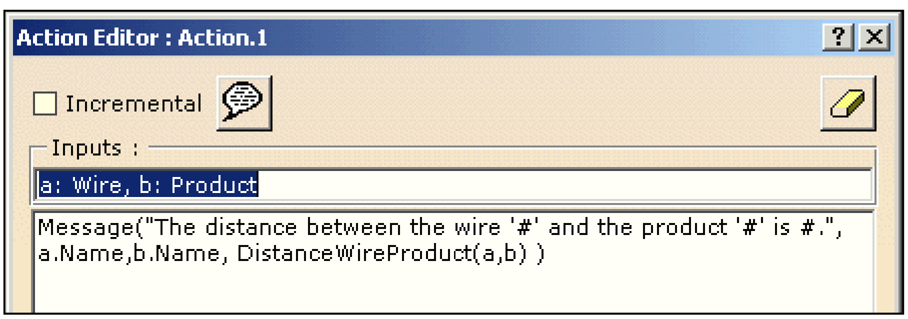


Running this rule displays the following message:



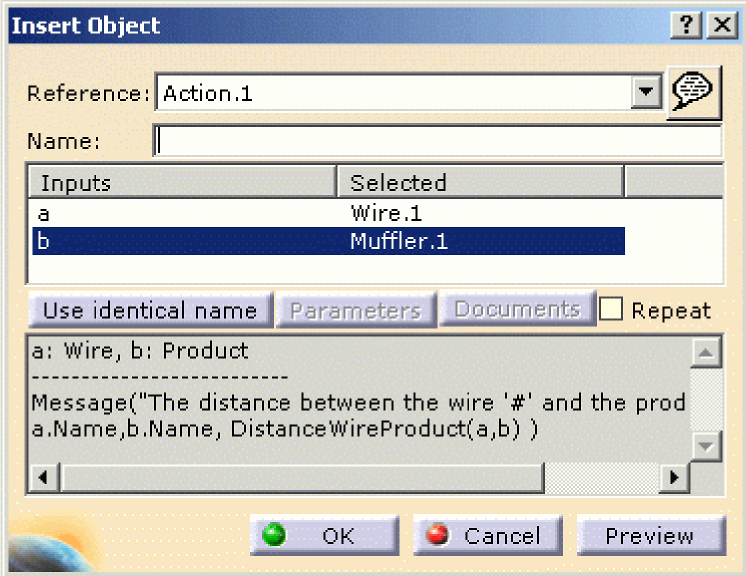
Example 4

Still in Knowledge Advisor, an action can be defined using the **DistanceWireProduct** user function, to know the distance between a wire and an object selected in the specification tree:

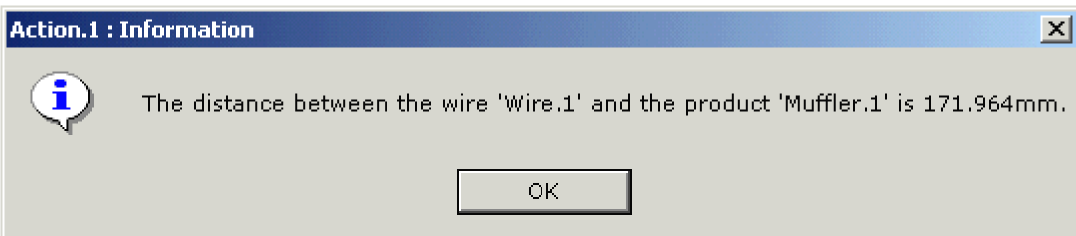


Run the action using the **Action.1** contextual menu:

select a wire and a product in the specification tree then validate.



This message displays:



VisualMode

Syntax

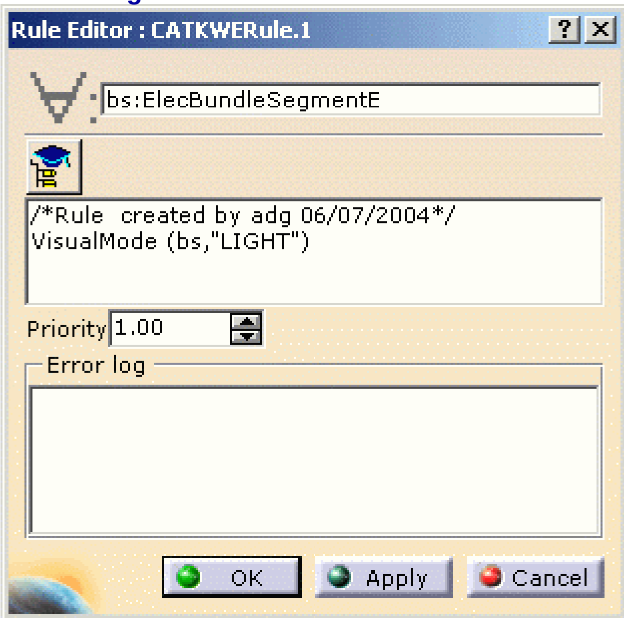
VisualMode (*BundleSegment: Feature, Visualization Mode (LIGHT/FULL): String*): **Boolean**

Description

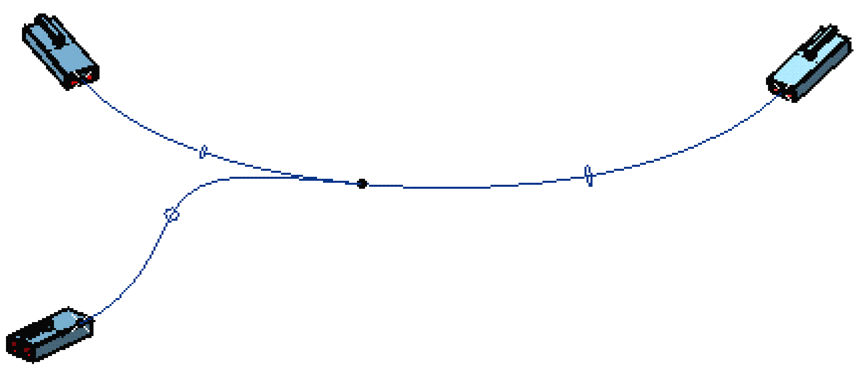
This function is useful to minimize the size of the harness in **LIGHT** mode. It simplifies the visualization, the curve and the diameter only are represented, the rib being deleted.

Example

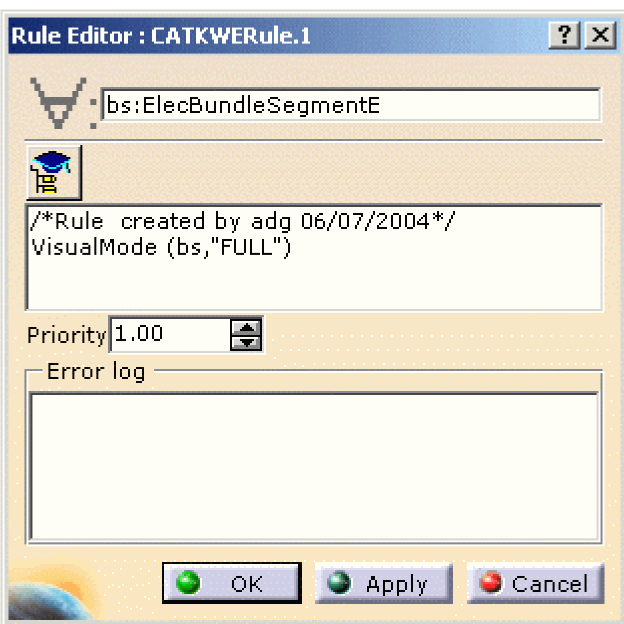
Create a rule in Knowledge Expert:



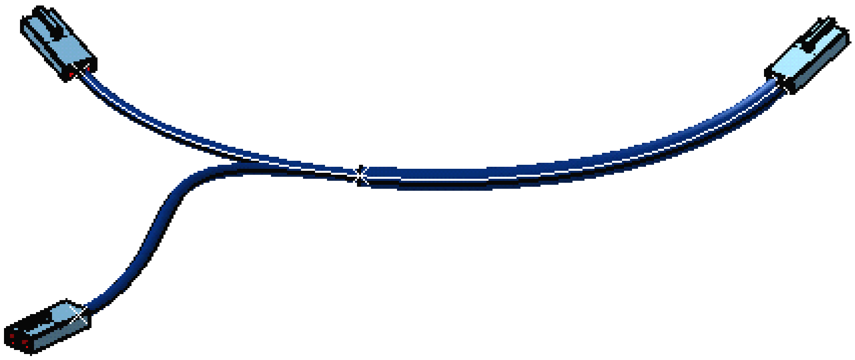
Running this rule displays the harness in **LIGHT** mode:



The rule is reversible: you can load a harness in **LIGHT** mode and reload the geometry by applying the rule with the **FULL** parameter:



The harness is displayed in FULL mode: the rib is recreated:



Creating a Formula

This task explains how to create a formula specifying that the external radius of a hollow cylinder is twice its internal diameter. Note that the radius of a sketch can be defined by a formula provided it is declared as a constraint.

Make sure the Relations option is active in the **Tools->Options...->Infrastructure->Part Infrastructure->Display** tab.

1. Open the KwrStartDocument.CATPart document.
2. Click the f(x) icon to display the *f(x)* dialog box . Make sure that the Incremental box is unchecked.

Method 1

- Double-click the PartBody\Sketch.1\Radius.3\Radius parameter in the parameter list. The Formula Editor is displayed.
- Enter the **PartBody\Hole.1\HoleLimit.1\Depth*2** relation in the formula field. Go to Tips and Techniques for information on how to manipulate parameters and formulas.
- Click **OK** in the Formula Editor.

Method 2

- Select the PartBody\Sketch.1\Radius.1\Radius in the parameter list.
- Click **Add Formula**. The Formula Editor is displayed.

- Enter the **PartBody\Hole.1\HoleLimit.1\Depth*2** relation in the formula field. Go to Tips and Techniques for information on how to manipulate parameters and formulas.

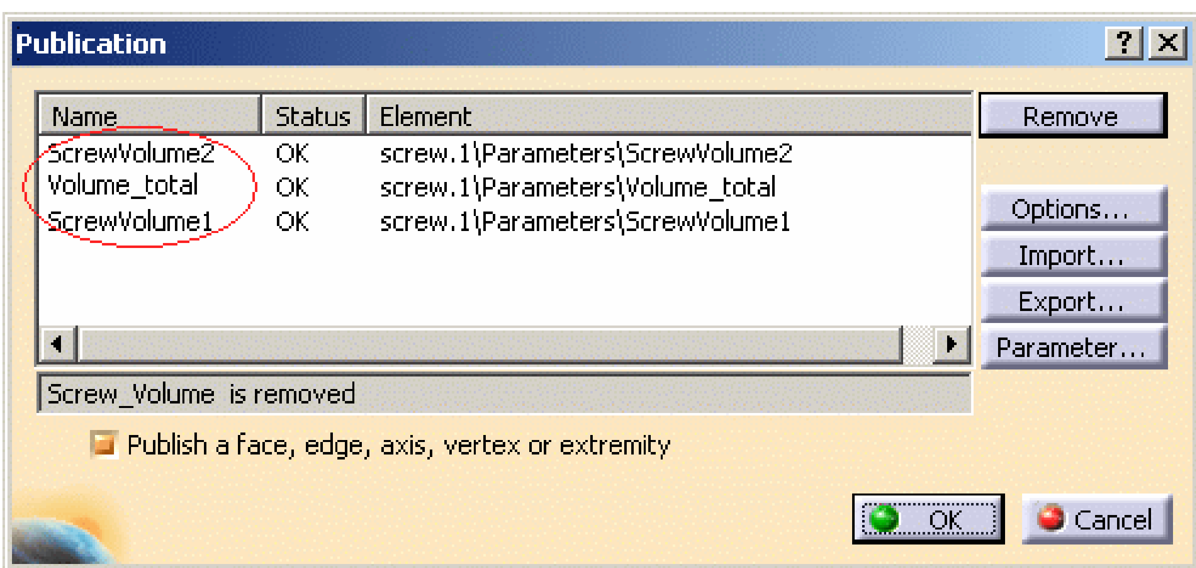
- Click **OK** in the Formulas Editor.

3. Click **Apply** to update the document.
4. Click **OK** to close the dialog box.

Creating Formulas based on Publications

This task explains how to create a formula based on publications in a CATProduct file.

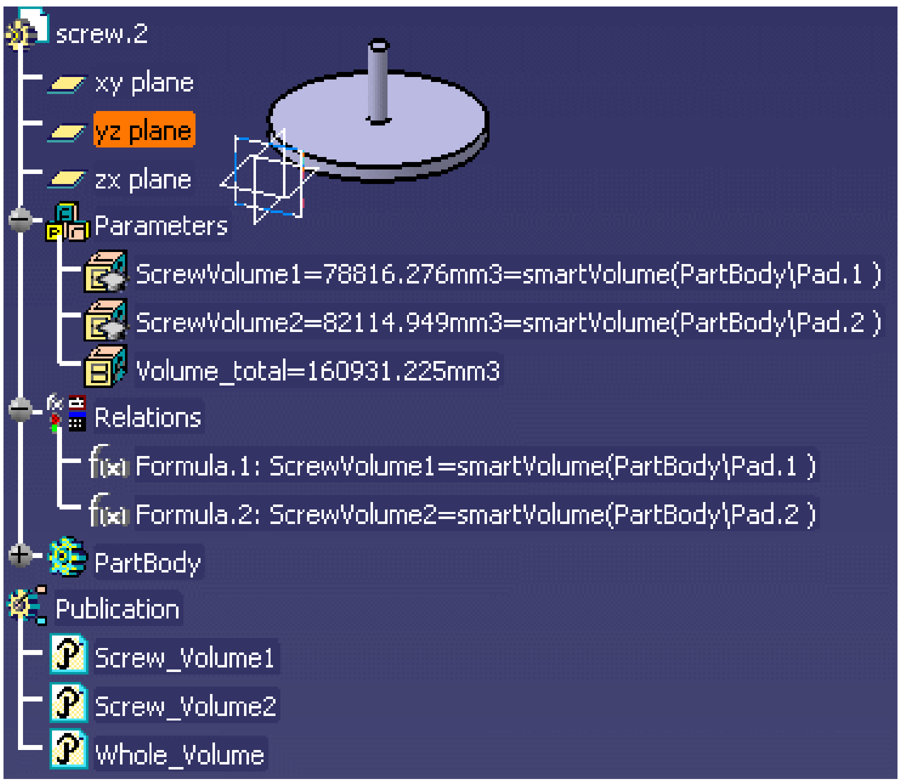
1. Open the Screw1.CATPart document.
2. Add a Volume parameter to the part. To do so, proceed as follows:
   - Click the icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Volume** and click the **New parameter of type** button.
   - In the **Edit name or value of the current parameter** field, enter the name of the parameter: **ScrewVolume1**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.
   - Enter the following formula by using the Dictionary to access the smartVolume operator: **ScrewVolume1=smartVolume(PartBody\Pad.1)**. Click **OK** three times.
3. Create another Volume parameter called ScrewVolume2 based on Pad.2.
4. Create another Volume parameter called Volume_Total. Click **OK** when done to exit the Formula editor.
5. Access the **Tools->Publication** menu, and select the 3 parameters that display below the Parameters node. Assign them new names (see graphic below):



6. Click **OK** when done. The parameters, the formulas, and the publications are created (see graphic below).

7. Save your file and close it.

8. Open the Screw2.CATPart document. Repeat the above steps (2 to 7).



9. Create a new product (**File->New** menu). Click **OK** when done.

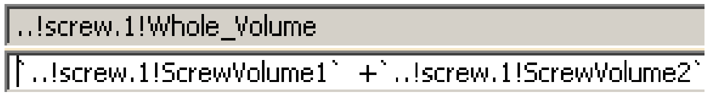
10. Select the **Insert->Existing Component...** command to insert Screw1.CATPart into the product.

11. In the File Selection window, select the Screw1.CATPart file that you have just saved. Click **Open**.

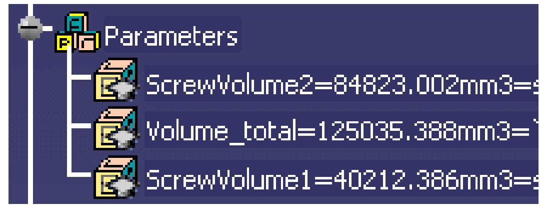
12. Create a formula that will compute the volume of the screw. To do so, proceed as follows:

- Click the Root product and click the  icon. The Formula Editor opens.

- Enter the following formula into the editor by selecting the publications in the specification tree:



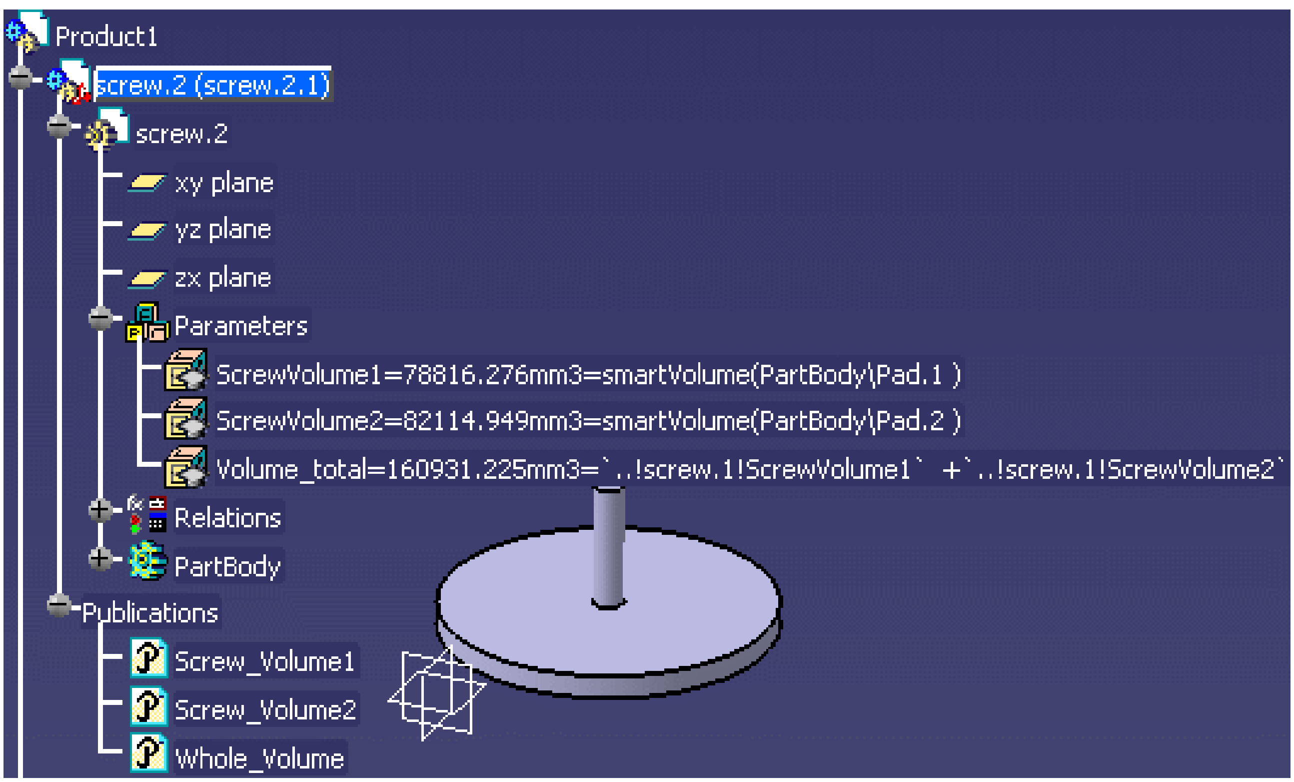
- Click **OK** when done. The screw volume displays below the parameters node.



13. Double-click screw.1 in the specification tree, right-click it, and select the **Components->Replace Component** command.

14. In the File Selection window, select the screw2.CATPart file that you have just saved and click **Open**.

15. Click **OK** in the Impacts On Replace window. The new screw is inserted into the product and its volume is computed.



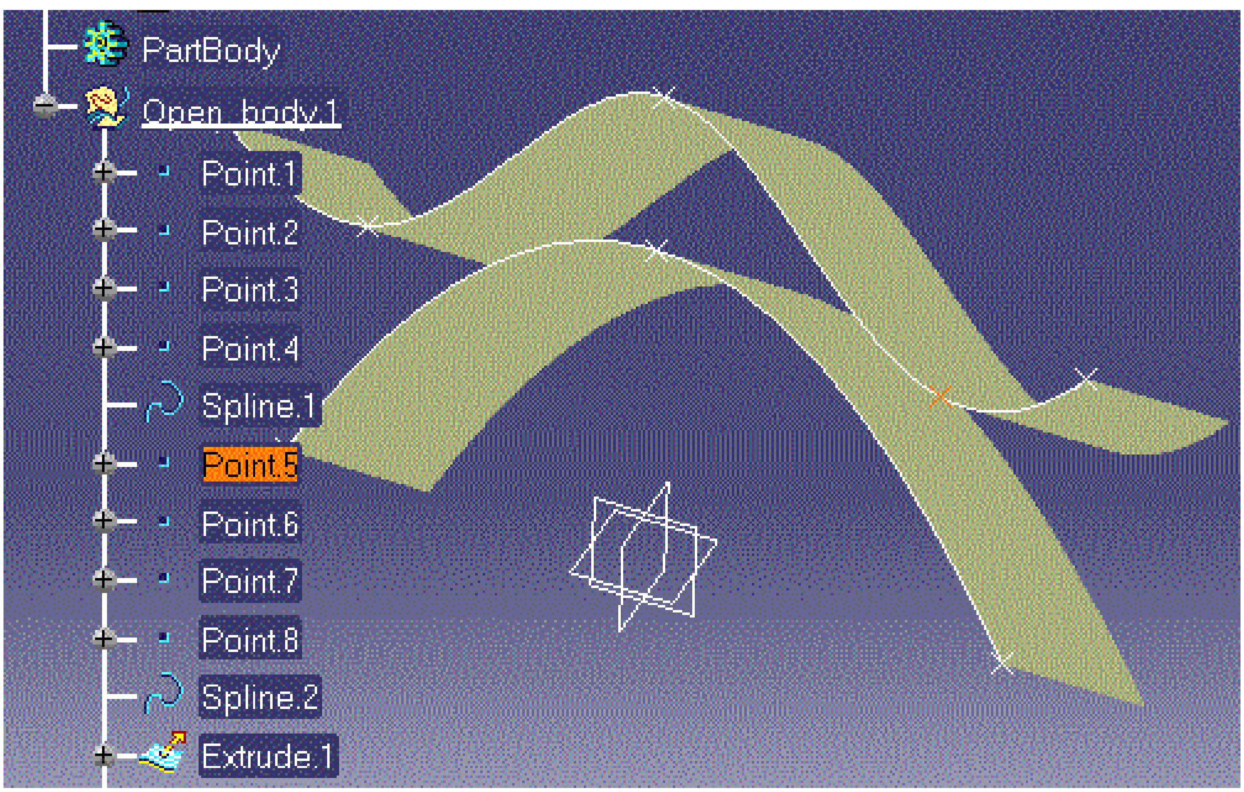
Specifying a Measure in a Formula

The purpose of this task is to explain how to specify that the value of a Length type parameter is equal to the curvilign abcissa of a point located on a curve.

Measures, i.e. values captured from the geometry area can be used in formulas. Here are some examples of measures that can be used in formulas:

- Distance between two points.
- Total length of a curve.
- Length of a curve segment - between a point and the origin or between a point and the curve extremity.
- Length of a curve segment - between two points.
- Area of an extruded surface.

1. Check the **Load extended language libraries** box in the **Tools->Options->General->Parameters and Measure->language** tab.
2. Open the KwrMeasure.CATPart document. The whole document has been created using the Generative Shape Design product. The Extrude.1 and Extrude.2 surfaces are extruded from the Spline.1 and Spline.2 curves. The point whose abscissa is to be measured is Point.5. The origin of the curve where Point.5 is located on is Point.8



3. Click the Formula icon. The *f(x)* dialog box is displayed.

4. Create the CurveLengthFromOrigin parameter.To do so, proceed as follows:

- Select the Length item with Single Value in the New Parameter of type list, then click New Parameter of type. The new parameter appears in Edit name or value of the current parameter.

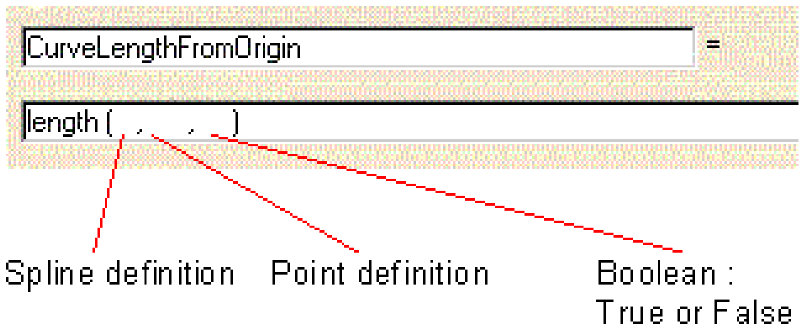
- Replace the Length.1 name with CurveLengthFromOrigin, and click Apply.

5. Specify that the value of CurveLengthFromOrigin is the abscissa of Point.5:

   a. Select the CurveLengthFromOrigin parameter in the parameters list, then click Add Formula. The Formula editor is displayed.

   b. Select the Measures item from the Dictionary list.

   c. In the list of measures, double-click the length(Curve,Point,Boolean) item. The length function is added to the Formula Editor.



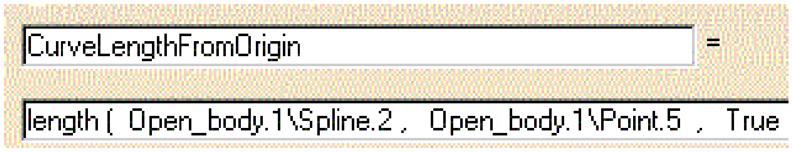
   d. Fill in the Formula editor field as indicated below.

      1. The three arguments are: a curve to be selected from the geometry area, a point to be selected from the geometry area and a boolean.

      2. Position the cursor where the first argument is intended to be typed. Then double-click the Spline.2 feature in the specification tree. The curve argument is added to the length definition.

      3. Position the cursor where the second argument is intended to be typed. Then double-click the Point.5 feature in the specification tree. The point argument is added to the length definition.

      4. Type a boolean for the third argument: True if the length is to be calculated from the origin, False if the length is to be calculated from the curve end.



5. Click **OK** to confirm the formula definition. You are back to the Formulas dialog box. The CurveLengthFromOrigin formula and value(47.5mm) are added to the parameter list.

e. Click **OK** to add the parameter as well as its formula to the document.

Using Geometry to Create a Formula

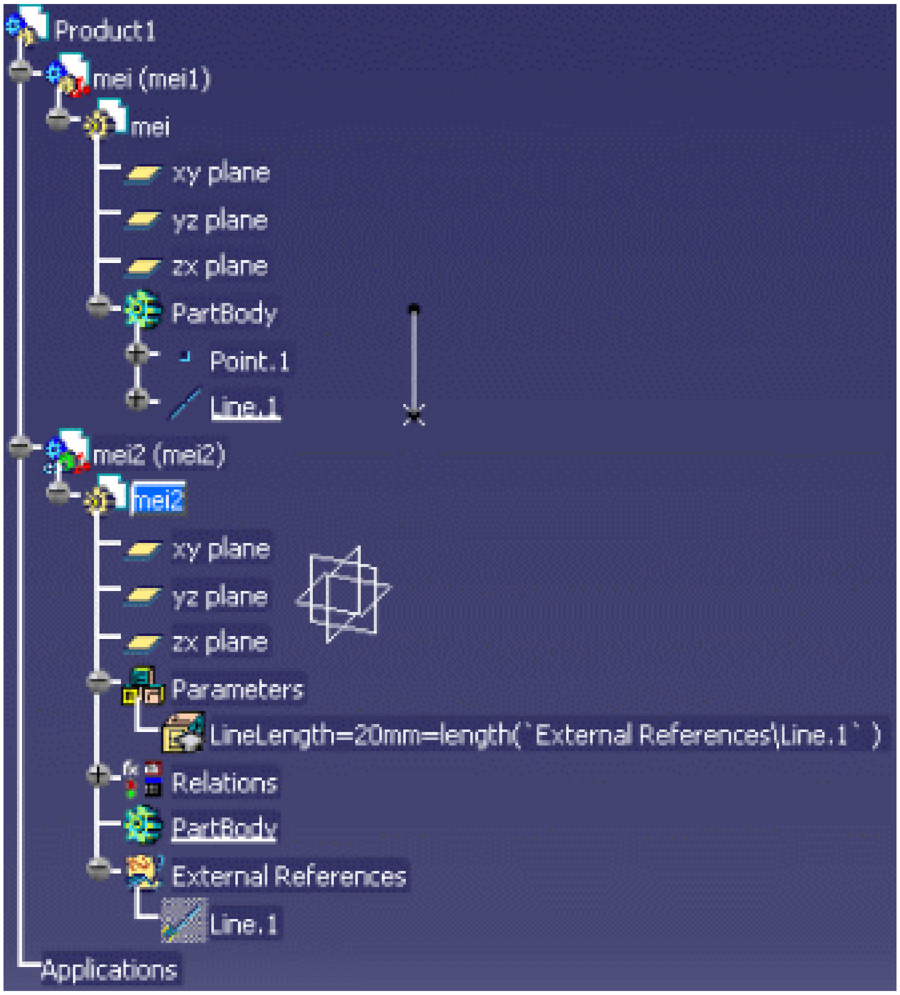
This task explains how to use geometry to create a formula.

Before carrying out this scenario, make sure you have checked the **Keep link with selected object** option (**Tools->Options-> Infrastructure->Part Infrastructure->General** tab.)

1. Create a Product and add 2 parts to this product.
2. From the **Start->Shape** menu, access the **Generative Shape Design** workbench.

   Select the first part.
3. Add a line to the part.
4. Select the second part and add a Length parameter to the part as well as a formula that will be based on the line. To do so, proceed as follows:

   - Click the  icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Legnth** and click the **New parameter of type** button.
   - In the **Edit name or value of the current parameter** field, enter the name of the parameter: **LineLength**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.
   - Enter the following formula by using the Dictionary to access the length(Curve,...) operator: LineLength=length(). Click **OK** three times.
   - Position the cursor between the parenthesis and select the Line. The External parameter selection window displays.
   - Select Line.1 and click OK. The formula displays as follows: LineLength=length(`External References\Line.1`)
   - Click **OK** when done. Click **Yes** in the Automatic update window.


Product1
mei (mei1)
mei
xy plane
yz plane
zx plane
PartBody
Point.1
Line.1
mei2 (mei2)
mei2
xy plane
yz plane
zx plane
Parameters
LineLength=20mm=length(`External References\Line.1`)
Relations
PartBody
External References
Line.1
Applications

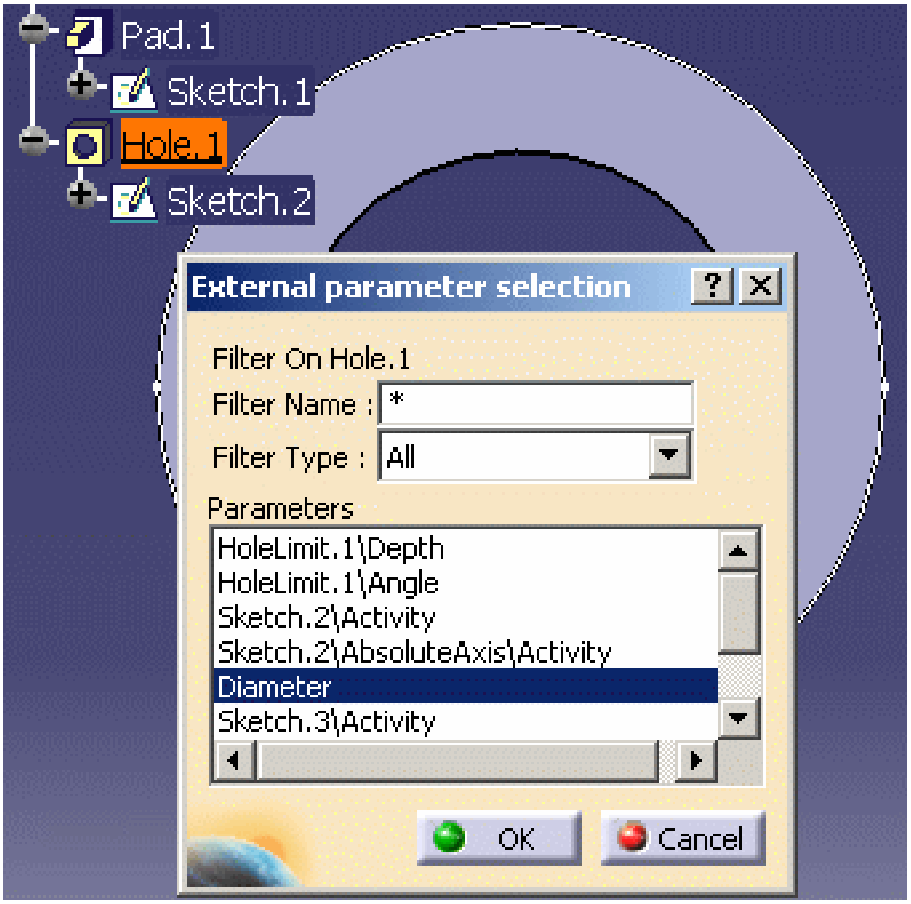
Referring to External Parameters in a Formula

This scenario shows how to use external parameters in a formula.

In a formula, you can use parameters defined in external documents. This works between any types of document. For example, in a CATPart document, you can specify a formula referring to parameters defined in a CATDrafting document. External parameters can also be used when working within an assembly.

Prior to carrying out this scenario, make sure that the **Keep link with selected object** option is checked (**Tools->Options...->Infrastructure->Part Infrastructure->General**).

1. Open the KwrStartDocument.CATPart document as well as the KwrImportParameter.CATPart document. Select the **Window->Tile Vertically** command from the standard menu bar. Both documents are displayed.
2. Make active the KwrImportParameter document. Click the icon to display the *f(x)* dialog box.
3. Create a parameter of Length type and click the Add Formula button. The formula editor is displayed.
4. In the KwrStartDocument specification tree, select the Hole.1 feature. The **External parameter selection** dialog box is displayed.

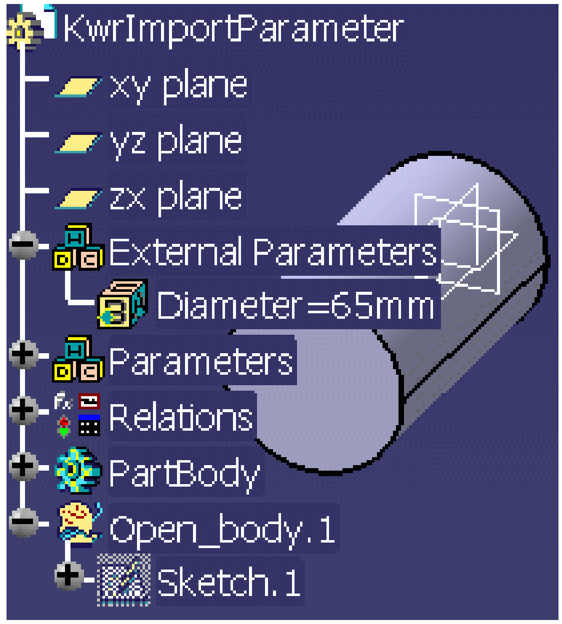


5. In the **External parameter selection** dialog box, select the Diameter object in the external parameter list. Then click OK. The Length.1 definition is carried forward to the formula editor. (Click the picture below to enlarge it.)

6. Complete the formula definition as indicated below:

   Length.1 = Diameter*0.45

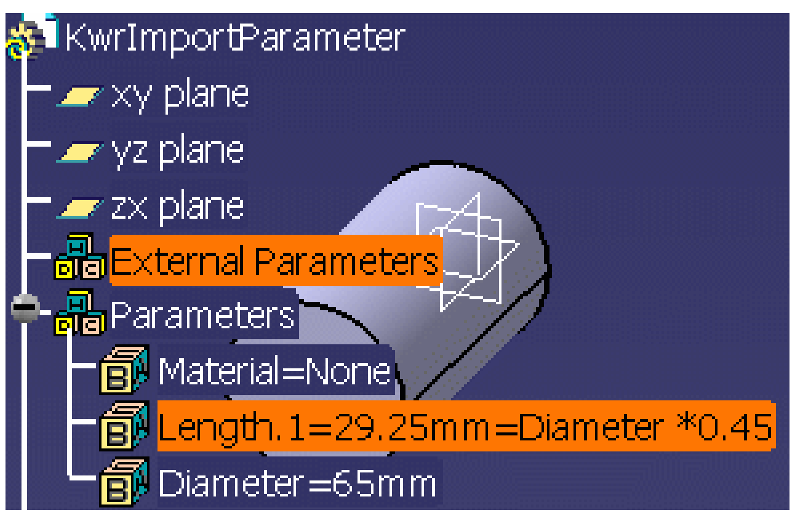
7. Click **OK** in the formula editor. You are back to the Formulas dialog box. In the parameter list, the Length.1 parameter value is modified according to the formula specified. In the KwrImportParameter specification tree, the External Parameters node is added. Expand this node to display the Diameter parameter.



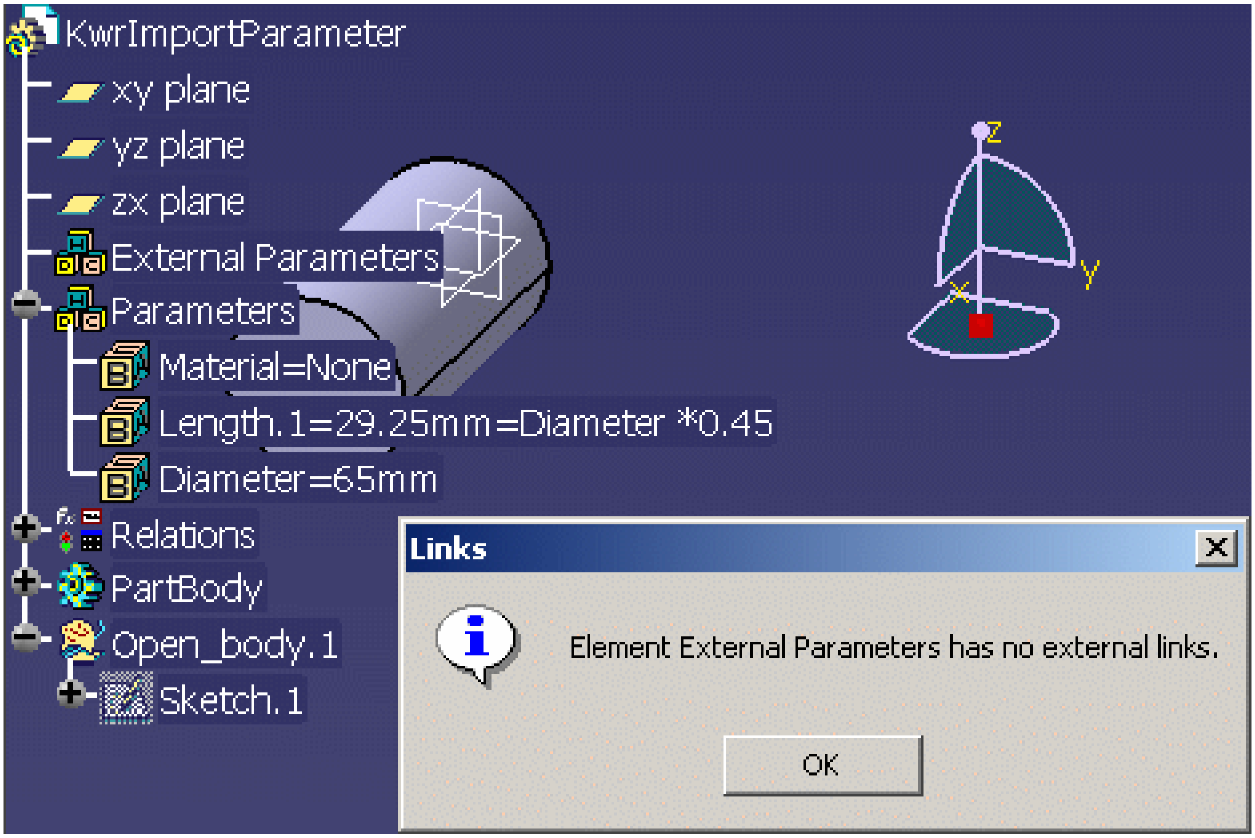
8. Click **OK** to add the formula to the KwrImportParameter.CATPart document and exit the dialog.

9. Select the Edit->Links command from the standard menu bar. The displayed dialog box confirms that there is a link between the KwrImportParameter\Length.1 object and the KwrStartDocument\PartBody\Hole.1\Diameter object.

10. Click **Isolate** in the Links dialog box, then click **OK**. In the KwrImportParameter.CATPart specification tree, the External Parameters node can no longer be expanded and the Diameter parameter is added below the Parameters node.



11. Select the **Edit->Links** command from the standard menu bar. A message box informs you that the active document has no external links.


KwrImportParameter
xy plane
yz plane
zx plane
External Parameters
Parameters
Material=None
Length.1=29.25mm=Diameter *0.45
Diameter=65mm
Relations
PartBody
Open_body.1
Sketch.1
Links
Element External Parameters has no external links.
OK

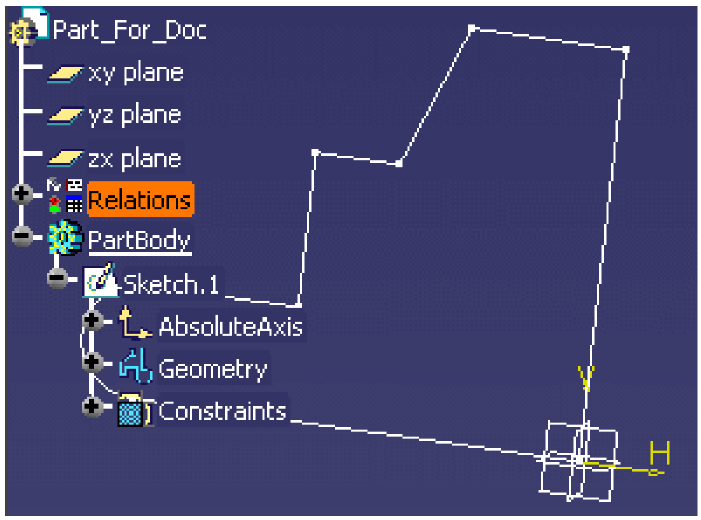
Using the Equivalent Dimensions Feature

This scenario explains how to use the Equivalent Dimensions Feature. The scenario described below is divided into the following steps:

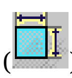
- The user apply constraints to an existing sketch.
- The user uses the Equivalent Dimensions feature to create a list of Length type parameters that will have the same value.

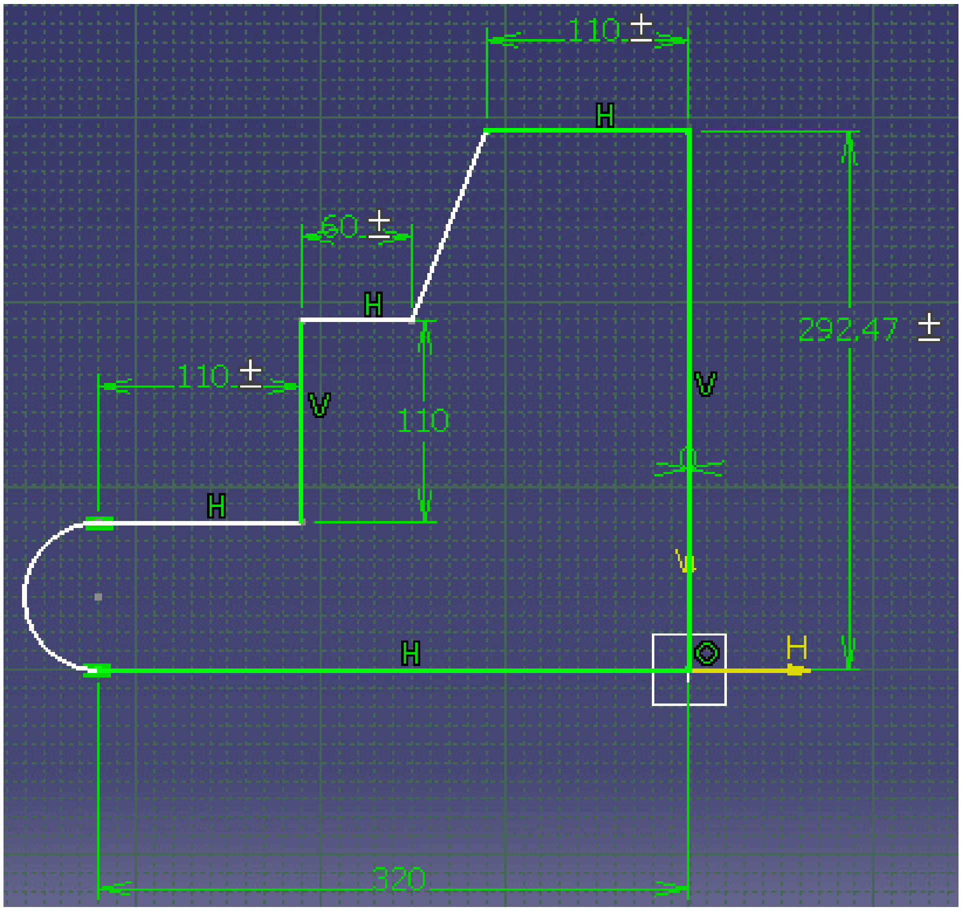
To know more about the Equivalent Dimensions feature, see Getting Familiar with the Equivalent Dimensions Interface.

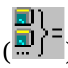
1. Open the KwrEquivalentDimensions.CATPart. The following image displays:



2. In the specification tree, expand the PartBody node and double-click Sketch.1 to access the sketcher.

3. Double-click the **Constraint** icon () to constraint some lines of the sketch (see graphic below).



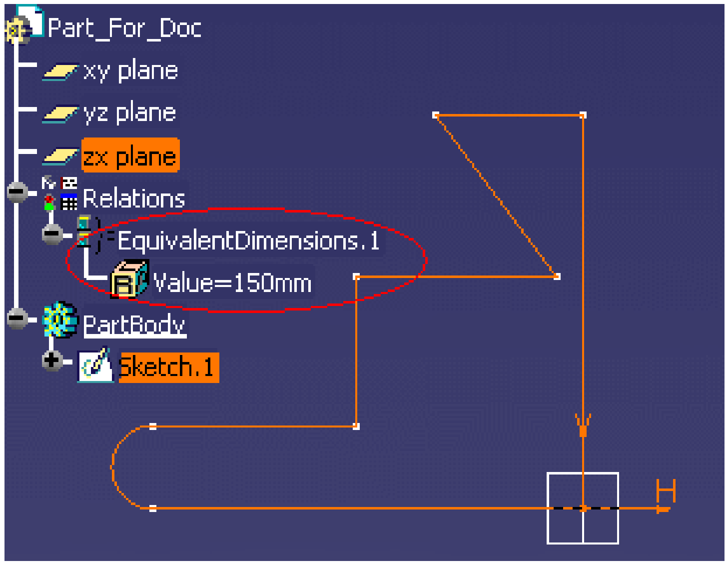
4. In the Knowledge toolbar, click the **Equivalent Dimensions** icon (). The Equivalent Dimensions Edition window displays.

5. Click the **Edit List...** button. In the opening window, use the arrow key to select the following parameters and click **OK** when done.

   - Length.34
   - Length.36
   - Length.37

6. In the Equivalent Dimensions Edition window, set the value to 150mm and click **OK**.

7. Exit the Sketcher. The sketch is modified accordingly and the EquivalentDimensions.1 feature displays below the Relations node.



8. Double-click Value=150mm twice in the specification tree. The **Edit Parameter** window displays.

9. Enter 140mm and click **OK**.

Associating URLs & Comments with Parameters and Relations

You can associate one or more URLs with user parameters and relations. This task is only meaningful when the active document contains user parameters and/or relations.

Adding URLs

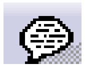
1. Click the icon (**Comment & URLs**). The **URLs & Comment** dialog box is displayed.
2. In the specification tree, select a parameter or a relation type feature.
3. If need be, select the **Edit** tab. Then click the **Add** button. The **Add URL** dialog box is displayed.
4. Enter a URL (http:\\www.foo.org for example) and a name. Click OK.
5. If need be, repeat this operation to add a new URL.
6. After you have finished entering all the required URLs, add a comment, then click **OK** to exit the dialog. The URLs and comment are added to the selected feature.

- When working with the URLs & Comment dialog box, please note that the icon located in the Knowledge Advisor toolbar enables the expert user to access a URLs & Comment dialog box where he can add, delete and modify the URLs. The icon available in the general toolbar is for the end-user only.
- To check that a user parameter or a relation has been assigned URLs, you just have to click the Comment & URLs icon and select the appropriate object in the specification tree.

Searching for a URL

When an object has been assigned a certain number of URLs, the Explore tab of the URLs &

Comment dialog box provides you with a way to search for a given URL.

1. Click the  icon and select an object (user parameter or relation) in the specification tree.
2. Select the **Explore** tab. The list of all the URLs assigned to the selected object is displayed.
3. Enter the name (or a sub string) of the URL to be searched for in the Search field. Then click Search. If the specified URL is found, "yes" is displayed in the Found column for every object containing a URL matching the search and only the first object to be found is highlighted.

Working with Design Tables

Select the Design Table icon to create a design table.

Introducing Design Tables
Getting Familiar with the Design Table Dialog Box
Creating a Design Table from Current Parameters Values
Creating a Design Table from a Pre-Existing File
Interactively Adding a Row To the Design Table External File
Controlling Design Tables Synchronization
Storing a Design Table in a PowerCopy

If you are already familiar with *CATIA* and only need a quick access to information, see the *CATIA* Knowledgeware Infrastructure - Tips and Techniques - Summary.

Introducing Design Tables

A design table:

- provides you with a means to create and manage component families. These components can be for example mechanical parts just differing in their parameter values.
- is a tool mainly intended to ease the definition of mechanical parts. It is provided to all *CATIA* users. But you will make the best use of it in a Knowledge Advisor application. A design table can be created from a *CATIA* document, the document data is then exported to the design table. It can also be applied to a document, the document data is then imported from the design table.
- is designed to drive the parameters of a *CATIA* document from external values. These values are stored in the form of a table either in a Microsoft ® Excel file on Windows™ or in a tabulated text file. When using a design table the trick is to associate the right document parameters with the right table parameters. The design table columns may not all correspond to your document parameters and you may decide to apply only part of the design table values to your document. By creating *associations*, you declare what document parameters you want to link with what table columns.
- becomes a more powerful tool when it is used with the Knowledge Advisor. You are provided with functions to read the design table parameters. These design table functions can be used when programming your checks and rules. Using these functions spares you all the association operations. To know more, click here.

Example

Screws are a good example of mechanical parts that can be described by a design table. To simplify, imagine they are all described by four parameters: the head width, the head height, the body width and the body height. The sets of four parameter values that can be assigned to a screw can be easily regrouped in a design table. This design table has as many columns as screw parameters and as many rows as sets of parameter values. In a design table, a set of parameter values is called a *configuration* and it is registered in a row.

The Excel Sheet Format (under Windows)

The values mentioned in the sheet cells have to be expressed in appropriate units. Otherwise, the right values won't be associated with the document parameters.

Only Excel sheets created with Excel 97 and subsequent versions are supported.

If no unit is mentioned within a cell:

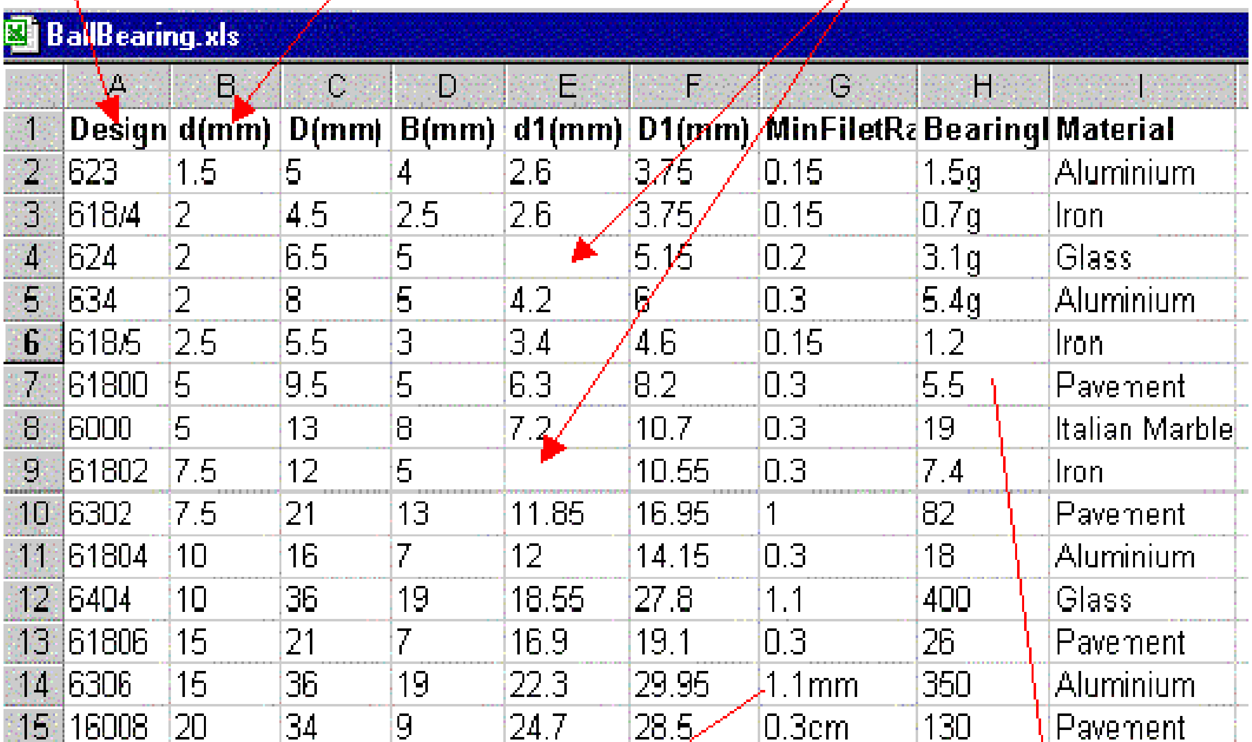
- the unit taken into account is the one mentioned in the first row
- and if no unit is specified in the first row, the unit taken into account is the relevant SI unit.

Here is an example of an Excel sheet:

column name

column unit

When a configuration which contains empty cells is selected, the parameters associated with the empty cells are not modified. This property enables you to modify parameters but only under certain conditions.



BallBearing.xls

	A	B	C	D	E	F	G	H	I
1	Design	d(mm)	D(mm)	B(mm)	d1(mm)	D1(mm)	MinFiletRa	BearingI	Material
2	623	1.5	5	4	2.6	3.75	0.15	1.5g	Aluminium
3	618/4	2	4.5	2.5	2.6	3.75	0.15	0.7g	Iron
4	624	2	6.5	5		5.15	0.2	3.1g	Glass
5	634	2	8	5	4.2	6	0.3	5.4g	Aluminium
6	618/5	2.5	5.5	3	3.4	4.6	0.15	1.2	Iron
7	61800	5	9.5	5	6.3	8.2	0.3	5.5	Pavement
8	6000	5	13	8	7.2	10.7	0.3	19	Italian Marble
9	61802	7.5	12	5		10.55	0.3	7.4	Iron
10	6302	7.5	21	13	11.85	16.95	1	82	Pavement
11	61804	10	16	7	12	14.15	0.3	18	Aluminium
12	6404	10	36	19	18.55	27.8	1.1	400	Glass
13	61806	15	21	7	16.9	19.1	0.3	26	Pavement
14	6306	15	36	19	22.3	29.95	1.1mm	350	Aluminium
15	16008	20	34	9	24.7	28.5	0.3cm	130	Pavement

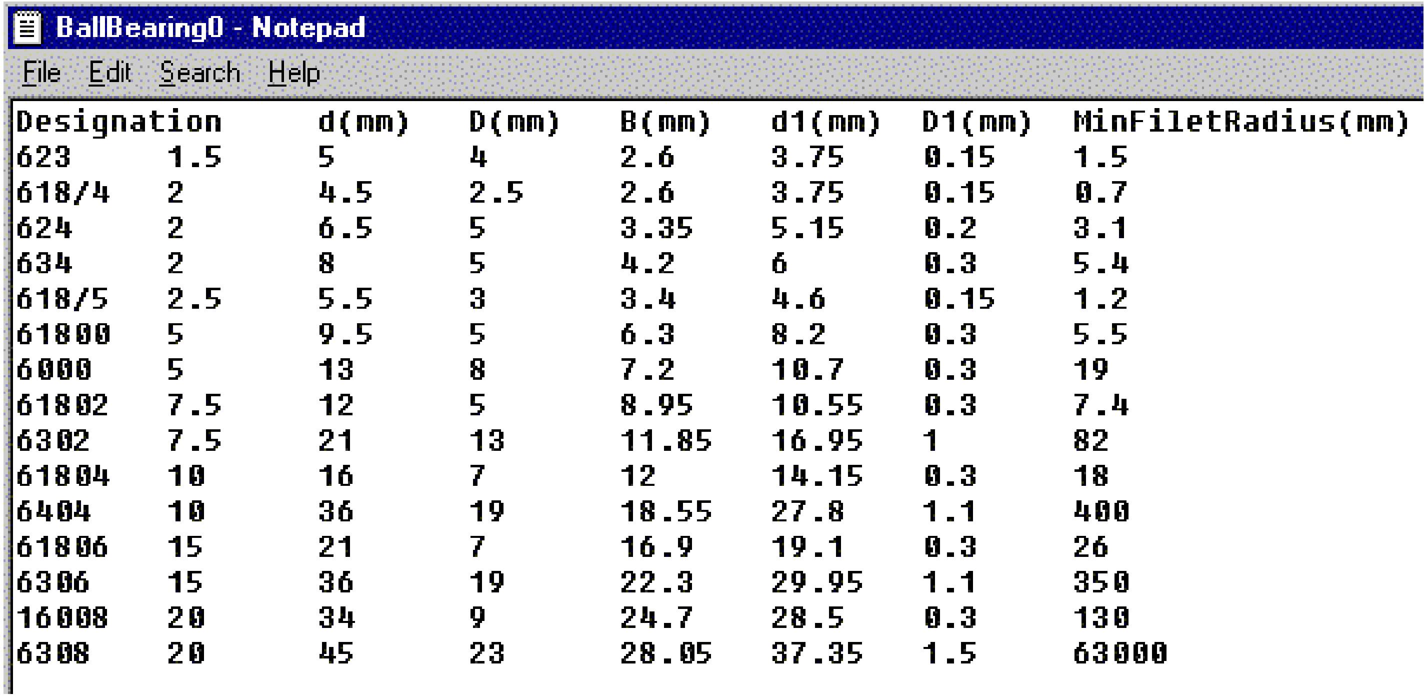
Within a given column, you can change the units.

Units can be specified in cells. No unit = SI

Note that it is highly recommended to choose the General format and not the Cells format in Excel.

The Tabulated Text File Format

Here is an example of a tabulated file format. You can use your favorite text editor to create this design table. Use the Tab key to skip from one column to the other. Unit rules are the same as for the Excel sheets.



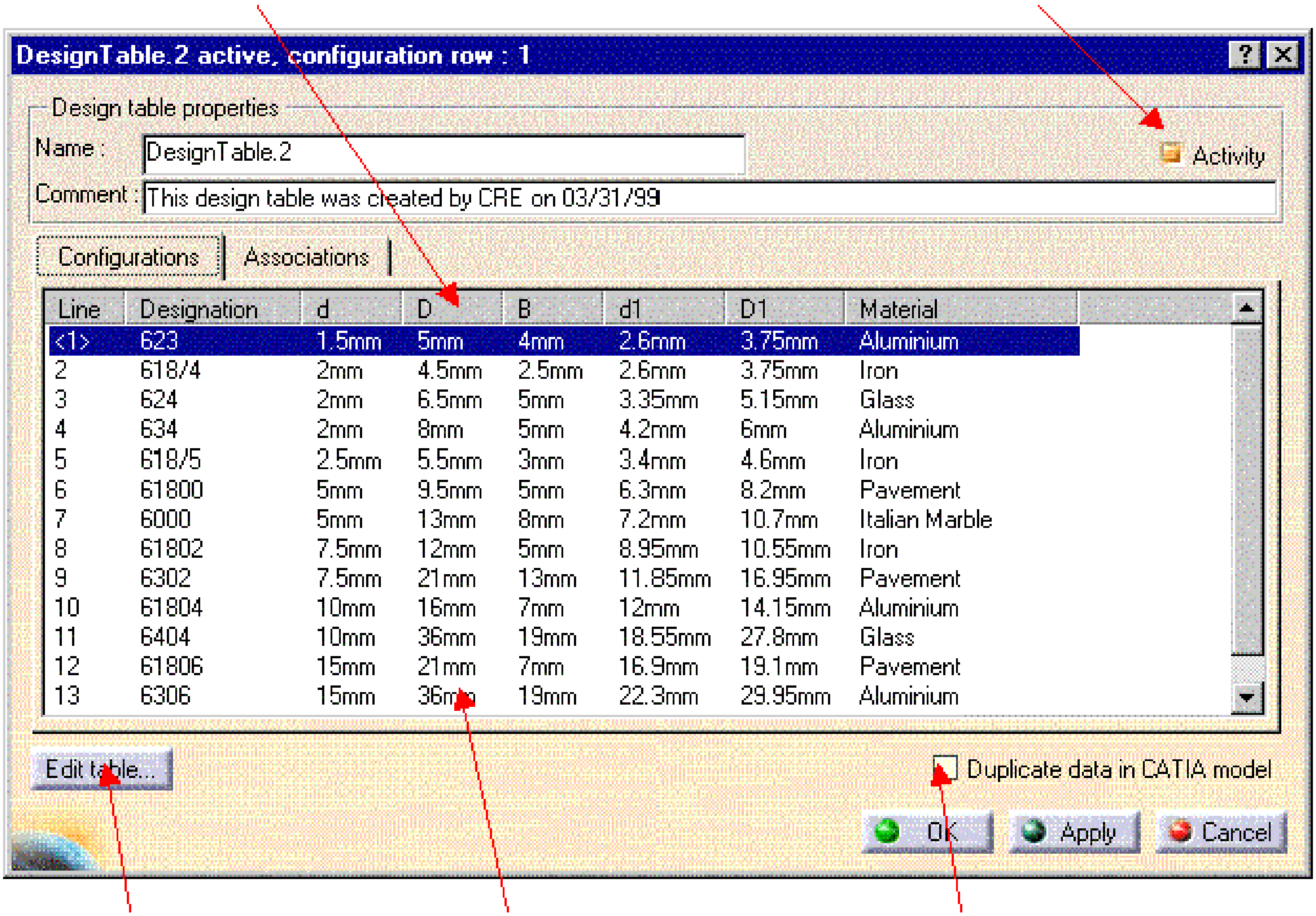
Designation		d(mm)	D(mm)	B(mm)	d1(mm)	D1(mm)	MinFiletRadius(mm)
623	1.5	5	4	2.6	3.75	0.15	1.5
618/4	2	4.5	2.5	2.6	3.75	0.15	0.7
624	2	6.5	5	3.35	5.15	0.2	3.1
634	2	8	5	4.2	6	0.3	5.4
618/5	2.5	5.5	3	3.4	4.6	0.15	1.2
61800	5	9.5	5	6.3	8.2	0.3	5.5
6000	5	13	8	7.2	10.7	0.3	19
61802	7.5	12	5	8.95	10.55	0.3	7.4
6302	7.5	21	13	11.85	16.95	1	82
61804	10	16	7	12	14.15	0.3	18
6404	10	36	19	18.55	27.8	1.1	400
61806	15	21	7	16.9	19.1	0.3	26
6306	15	36	19	22.3	29.95	1.1	350
16008	20	34	9	24.7	28.5	0.3	130
6308	20	45	23	28.05	37.35	1.5	63000

Under UNIX, it is possible to change the default design table editor. To do so, type:
export CATTextEditorDT=... (indicate the path of the editor.)

The *CATIA* Design Table

Once it has been read and processed by *CATIA*, the design table looks something like this:

No units in column   Check box to modify the activity

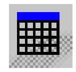


Displays the design table raw data.

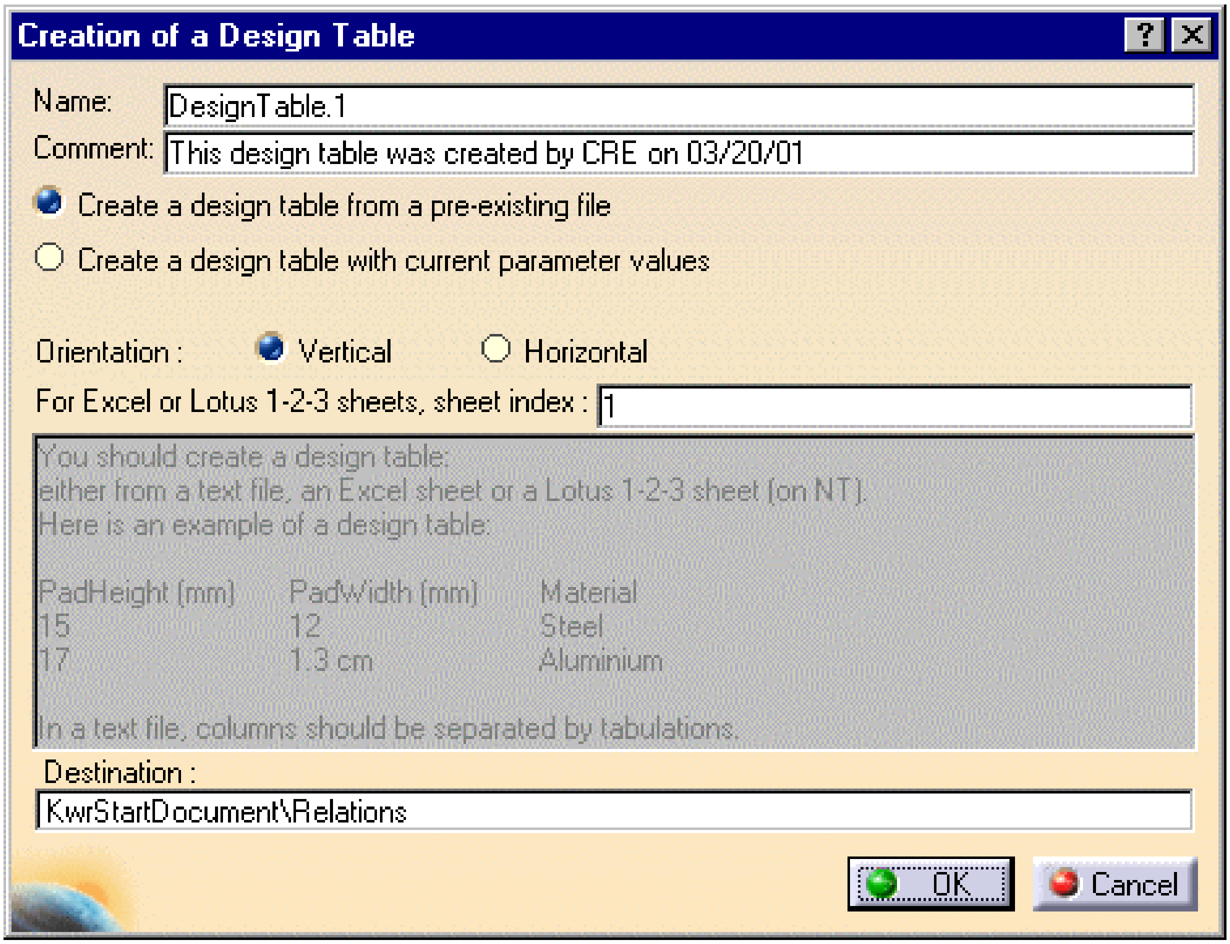
Values with units

Duplicates the design table external data into the *CATIA* document. Check this box whenever you intend to re-access your design table on another platform.

Getting Familiar with the Design Table Dialog Box

Here is the dialog box sequence you get onscreen when you click the  icon in the standard toolbar.

Creation of a design table



"Create a design table from a pre-existing file" *check box*

Check this option whenever you want to create a design table from the values of an external file. In this case, the created design table is made up of:

- either only the columns whose name is a document parameter name. If the external file contains a "Length" column but no such "Length" parameter is defined in the document, the "Length" column will not appear in the created design table. This is the "automatic" association process.
- or only the columns that have been associated one-by-one with a document parameter. If the external file contains a "Length" column but no so-called parameter in the document, you can choose to associate the "Length" column of the external with a parameter of the same type (a sketch radius for example).

"Create a design table with current parameter values" *check box*

Check this option whenever you want to create a design table from a subset of the document parameters. You just have to select among all the document parameters the ones you want to be included as columns in the design table. In this case, the created design table only contains a single row.

*The **Orientation** check boxes*

These options allow you to choose the design table orientation. A vertical orientation is recommended when the design table contains many parameters.

*The **sheet** index*

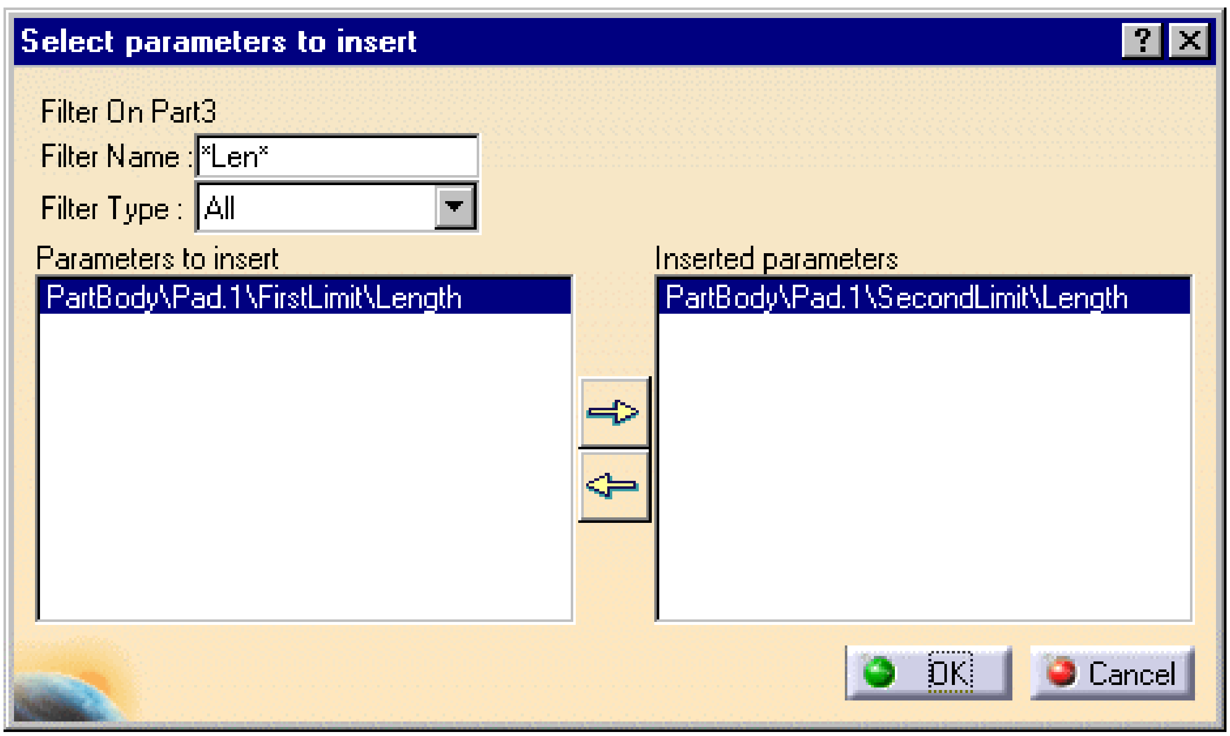
From Version 5 Release 7, you can specify an Excel or Lotus sheet number.

*The **Destination** field*

All knowledgeware relations such as design tables, rules, checks or formulas, are created by default below the Relations node. Creating a relation below a given feature may help you organize your document. To specify a destination, select the default destination in the Destination field, then click the feature intended to be the new destination either in the specification tree or in the geometry area.

Selection of the parameters to insert

This dialog box pops up when you check the "Create a design table with current parameter values" check box.



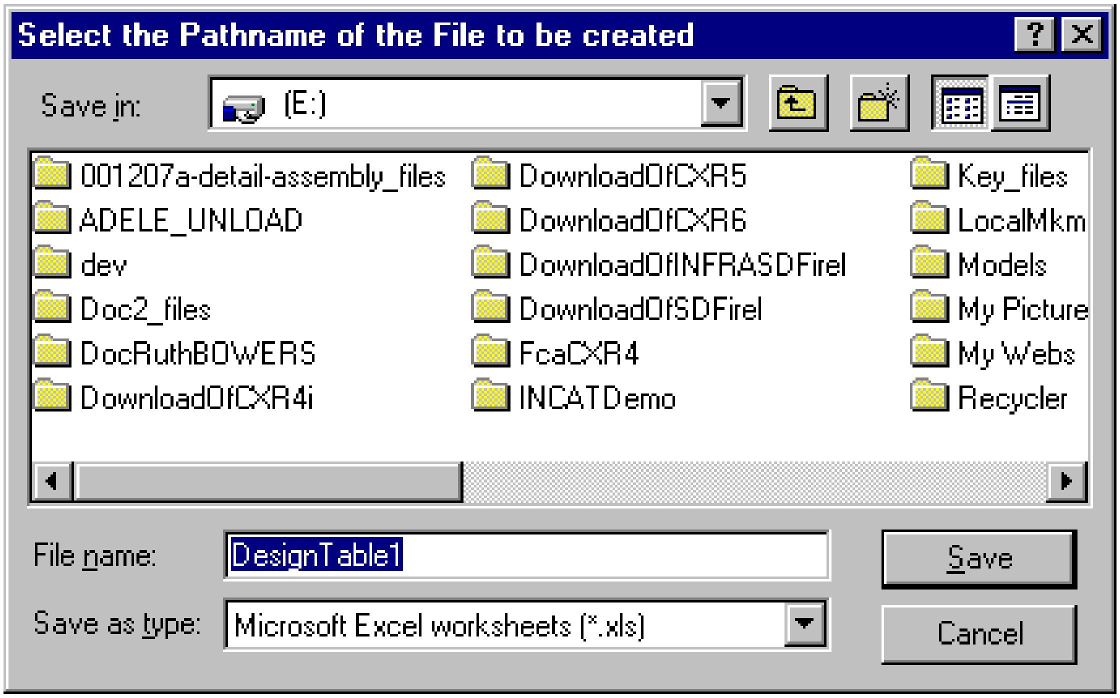
There are two ways to restrict the list of parameters to be displayed in the 'Parameters to insert' list. You can use the:

1. *Filter Name* field

Use the * character to specify any string to be included in a parameter name. Specifying *Len* will display in the "Parameters to insert" part of the dialog box all the parameters having the Len substring in their name.
2. and the *Filter Type* field.

When you click OK in the dialog box above, the "Select the pathname of the file to be created" panel is displayed.

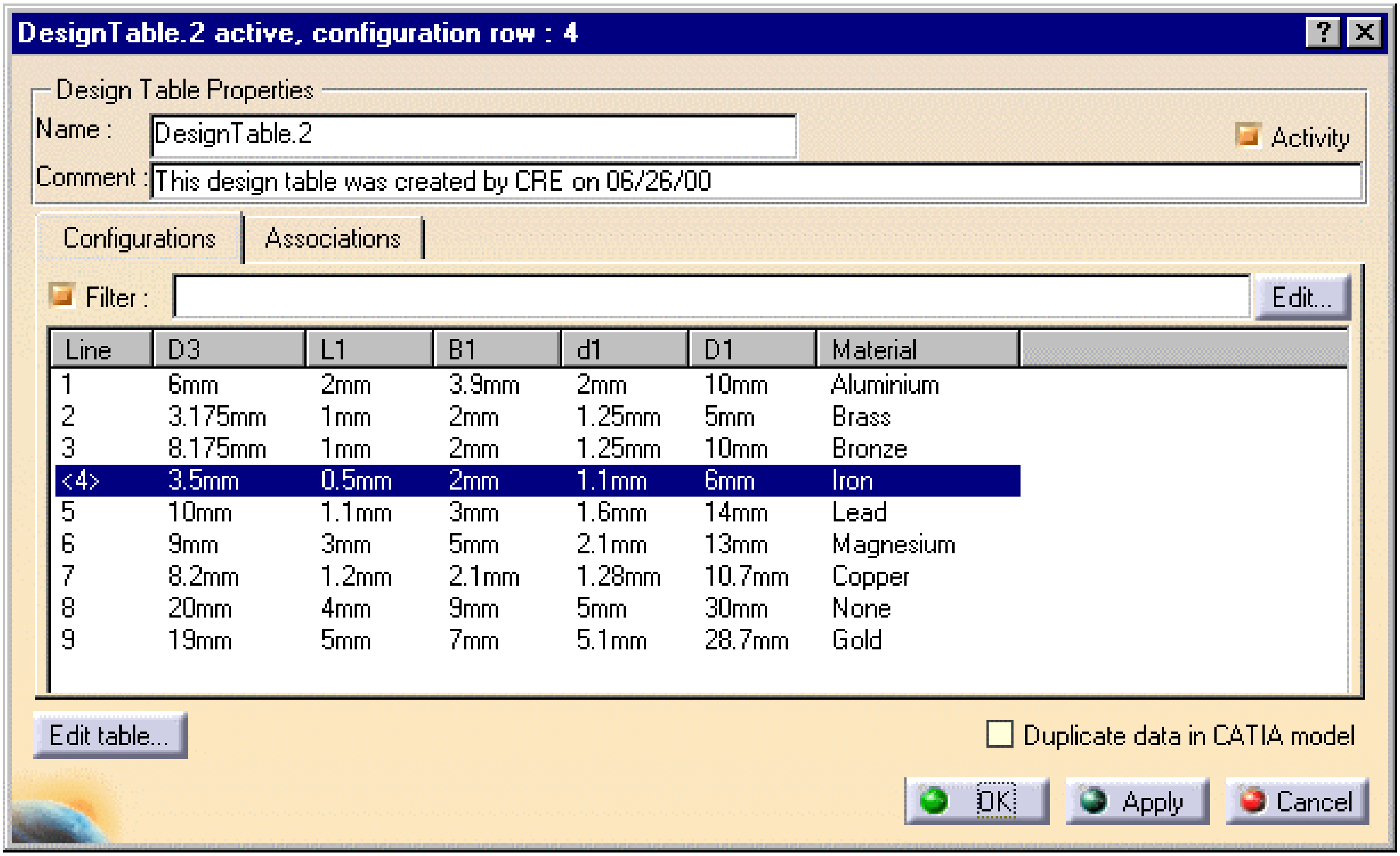
Selection of the file to be created



Use this dialog box to specify the .xls (Windows) or .txt file to be created. Specify the .xls extension when filling out the 'File name' field. Then click Open to display the design table dialog box.

Design table dialog box

The 'Configurations' tab



The current configuration as well as its number (< *configuration number* >) are highlighted. To change the current configuration, you just have to click the new configuration in the design table.
A single row design table is created when you generate a design table with the current parameter values.

- ***The Filter***
  The filter is a means to help you query for a configuration meeting specific criteria. Click the "Edit... " button to display the "Design Table Request Editor". See Using the Dictionary for information on how to use the syntax provided by the dictionary.
  In a query, you can specify a condition referring to the design table parameters as well as the parameters external to the design table.

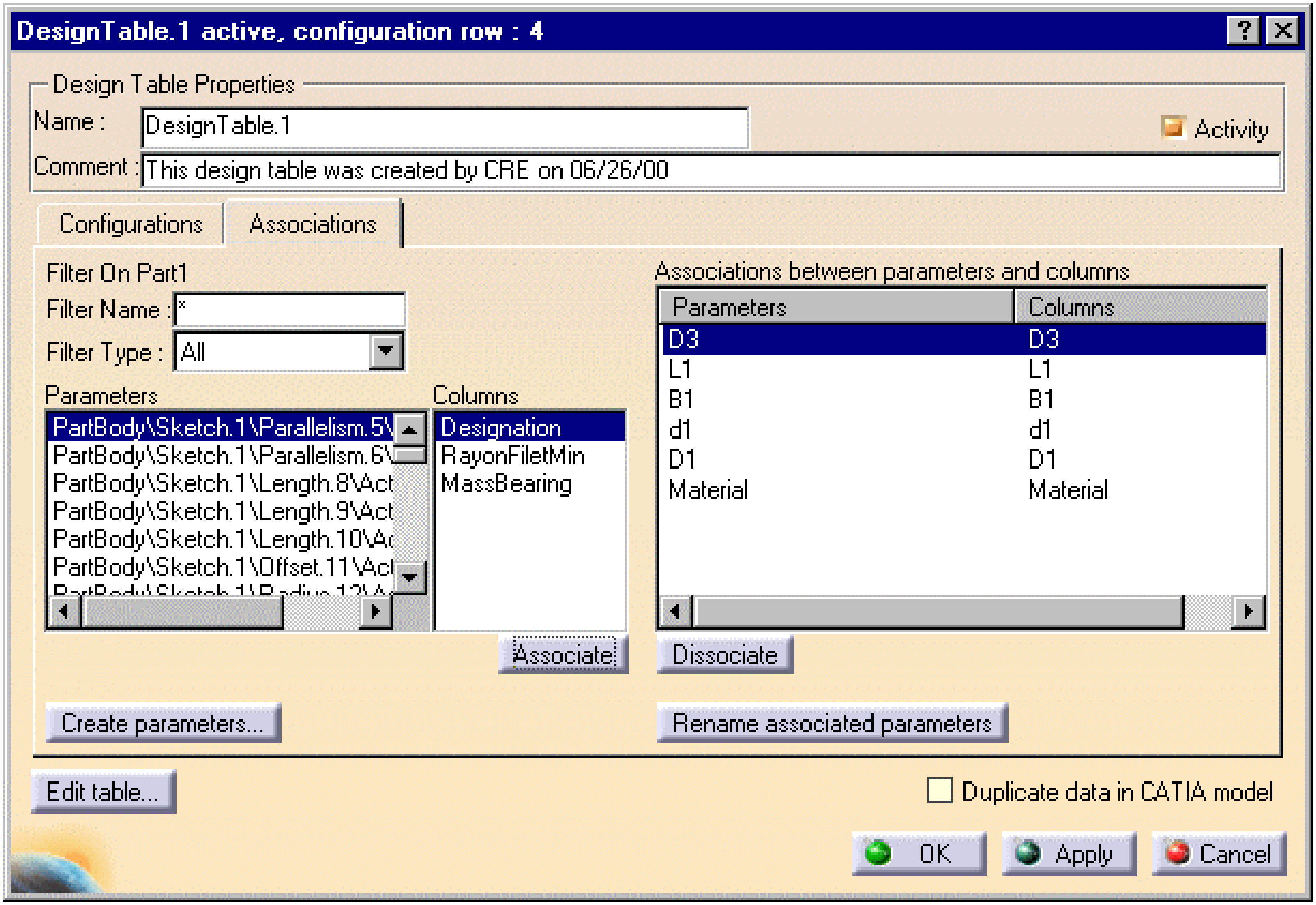
- ***The "Activity" check box***
  A design table is created active by default. The activity check box provides you with a way to deactivate the design table to be created.

- ***The "Edit table..." push button***
  Click this button to display the edit table to be created. Depending on whether you have selected a .xls extension or not, you will launch a Microsoft Excel application or your default text editor for a .txt file.

- **The "Duplicate data in CATIA model" check box**
  Check this box whenever you intend to reuse your document on an operating system different from the one used to create the design table. That way, your design table data will be duplicated into your document.

The "Associations" tab

This tab provides you with a way to associate the document parameters with the columns of the external design table. The left part of the dialog box allows you to associate parameters with the design table columns while the right part displays the list of associated parameters.



- ***The "Create parameters..." push button***
  When a parameter is referred to in the design table but has not been created in the document, clicking this button allows you to create a parameter in the document and associate it with the right column of the design table.

- ***The "Rename associated parameters" push button***

  If a parameter does not have the same name as the column it is associated with, you can rename this parameter so that it has the same name as the column. Clicking the "Rename associated parameters" push button displays a dialog box which asks you whether you want to rename all the parameters or only a few of them.

Creating a Design Table from the Current Parameters Values

A design table is a feature that you create from your document parameters or from external data. No matter the existence of external data, you must **create** the design table in *CATIA*. There are two ways to create a design table:

- From the current parameter values
- From a pre-existing file.

The scenario described below explains how to proceed in the first case. The design table creation process includes the following steps:

a. Create a table from the document parameters.

b. Select the parameters to add to the design table.

c. Specify a file to contain the generated design table.

d. Edit the generated *CATIA* design table.

e. Apply the design table to your document.

For information on how to use the different dialog boxes related to the design table, see The Design Table Dialog.

1. Open the KwrStartDocument.CATPart document.
2. Click the Design Table icon in the standard toolbar. The Creation of a Design Table dialog box is displayed. See The Design Table Dialog for further information.
3. If need be replace the default name and comment for the design table.
4. Check the **Create a design table with current parameter values** option.
5. Click OK. The **Select parameters to insert** dialog box is displayed.
6. In the Parameters to insert list, select the PartBody\Pad.1\FirstLimit\Length and the PartBody\Pad.1\SecondLimit\Length items. Then click the right arrow to add both items to the Inserted parameters list.
7. Click **OK**. A file selection box is displayed.

8. Specify the pathname of the design table to be created. Click **OK** in the file selection dialog box.

   The design table feature is added to the specification tree and a dialog box displays the newly created design table. This design table contains only one configuration. By default it is active.

   If the file specified already exists, the Creation of a Design Table dialog box is re-displayed as well as a message box asking you whether you want to overwrite the existing file.

9. Click **Edit table...** to start an Excel application (under Windows) or open the text editor under Unix.

   Replace the PartBody\Pad.1\FirstLimit\Length parameter value with 80mm.

10. Save your Excel or .txt file and close your application. Some information messages are displayed in a dialog box warning you about events related to the design table. Click Close.

11. Click **Apply** into the *CATIA* design table dialog, the document is updated as well as the *CATIA* design table. Click OK to exit the dialog and add the design table to the document.

Creating a Design Table from a Pre-existing File

A design table is a feature that you create using your document parameters or external data. No matter the existence of external data, the design table must **created** in *CATIA*. There are two ways to create a design table:

- Using the current parameter values
- Using a pre-existing file

The scenario below describes how to proceed in the second case. Here are the main steps to follow:

a. Select the pre-existing file containing the raw data.
b. Create the associations between the document parameters and the external table columns. You can choose to create these associations automatically.
c. Edit the generated *CATIA* design table.
d. Select a configuration in the generated design table. You can modify the default configuration proposed by *CATIA*.
e. Apply the design table feature to your document.

For information on how to use the different dialog boxes related to the design table, see The Design Table Dialog.

It is now possible to select a Design Table external file from a VPDM (ENOVIA VPM V5, ...). To do so, make sure you have enabled the desired environment in the Document Environments field (**Tools->Options->General->Document**.) Your documents will be accessible via the Document Chooser.

1. Open the KwrStartDocument.CATPart document.
2. Click the Design Table icon ( ) in the standard toolbar.
   The **Creation of a Design Table** dialog box is displayed. Enter a name (DesignTable1 for example) and a comment.
3. Check the **Create a design table from a pre-existing file** option. Click **OK**.
4. Select the KwrBallBearing.xls file, and click Open. A dialog box asks you whether you want to perform automatic associations between the design table columns and the document parameters which have the same name.
5. Click **Yes**. The **Material** parameter is the only one which is common to the document parameters and to the external design table. A multi-row design table is created. The '<' and '>' symbols denote the current configuration.

6. Select the configuration you want to apply to the document (line 4 for example). Click **Apply**.

   The Iron parameter value is displayed in the specification tree.

7. Click **OK** to end the design table creation.

The scenario below illustrates how to create a design table by associating one by one the document parameters with the input file columns.

1. Open the KwrStartDocument.CATPart document.

2. Click the Design Table icon in the standard toolbar.
   The "Creation of a Design Table" dialog box is displayed. Enter a name (DesignTable2 for example) and a comment.

3. Check the Create a design table from a pre-existing file option. Click **OK**. A file selection panel is displayed.

4. Select the KwrBallBearing.xls file. Click **Open**. The Automatic associations dialog box is displayed.

5. Click **No**. The design table dialog box informs you that there is no associations between parameters and columns.

   Now, you have to associate one by one the document parameters with the design table columns.

6. Click the **Associations** option. The table design dialog box now displays side by side the document parameter list and the input file columns.

7. In the Parameters list, select the PartBody\Hole.1\Diameter item. In the Columns list, select the d1 parameter. Then click **Associate**. A parameter couple is now displayed in the Associations between parameters and columns list.

8. Repeat the same operation for the Material parameter.

Selecting a parameter or an association in the list highlights the corresponding values in the geometry area.

The parameter list can be filtered:

- By clicking on a feature (either in the specification tree or in the geometry area). All the parameter values of the selected feature (and children) are highlighted in the geometry area. The parameter list displays only the parameters of the selected features (and children).
- By specifying a string in the Filter Name field. For example, typing *ength* displays all Length parameters
- By specifying a type in the Filter Type field.

The **Create parameters...** button allows you to create automatically parameters and associations for items of the Columns list. The Rename associated parameters button replaces the parameter name with the column name.

9. Click **OK** to end the DesignTable2 creation dialog.

The DesignTable2 feature is added as a relation to the specification tree. Double-click DesignTable2 in the specification to edit the table. By default, the configuration 1 is applied to the document. A new material (Aluminum) is applied to the document and the hole diameter is modified.You can select another configuration and apply it to your document.

Interactively Adding a Row To a Design Table External File

The task described below explains how to add a row to a design table external file. The scenario is divided into the following steps:

- The user opens the CATPart file and inserts the design table
- The user deactivates the design table and creates a new configuration
- The user adds the configuration to the design table external file
- The user activates the design table and implements the new configuration

This new function enables the user to add a contextual menu on design table feature (in the tree) which appears only:

- If the design table is deactivated
- If the design table external file exists and is read/write
- If at least one parameter is associated.

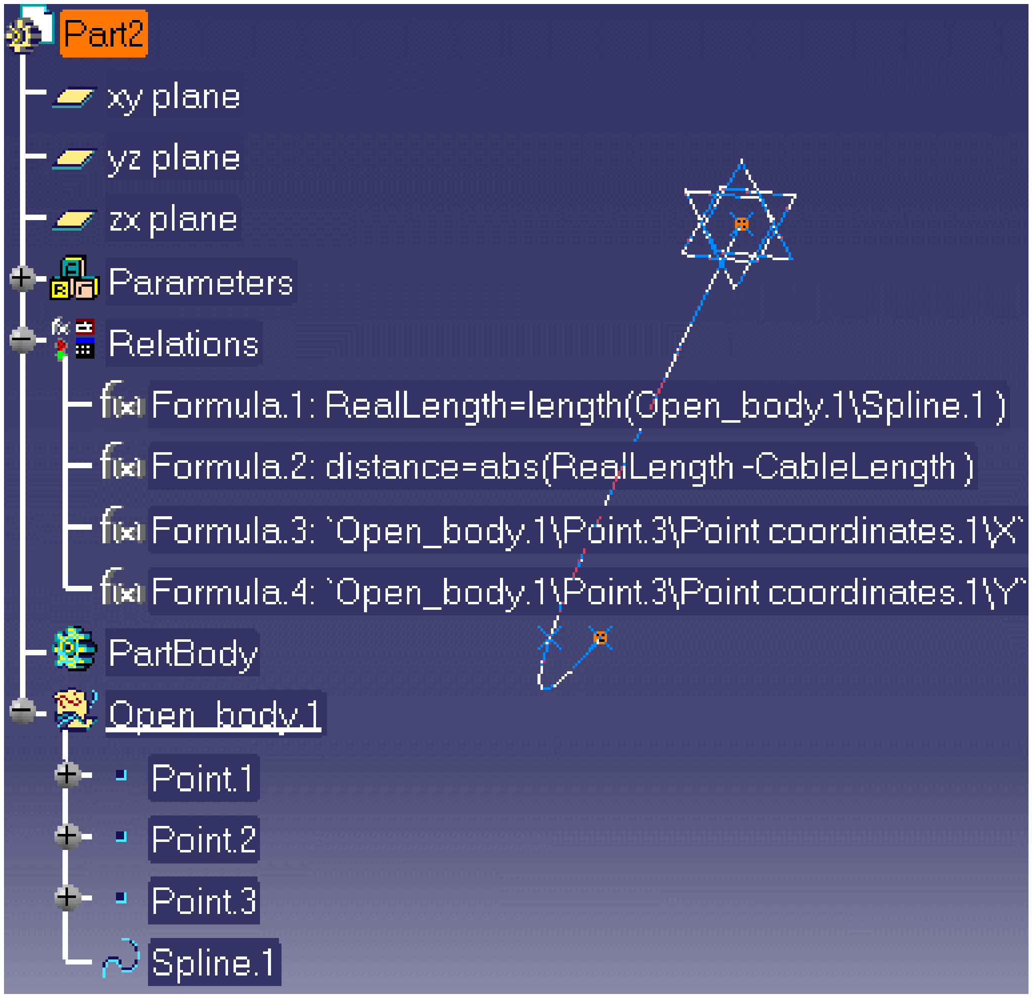
The behavior of this command is to add a row at the end of the design table file with associated parameters values. For not associated columns, an empty cell is added.

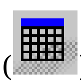
To carry out this scenario, the user will need the following files:
KwrAddARow.CATPart
KwrAddARow.xls

Note that this task can only be performed in an english environment.

Prior to carrying out this scenario, make sure the **With value** and **With formula** options are checked in the **Tools->Options->General->Parameters and Measure->Knowledge** tab.

1. Open the KwrAddARow.CATPart file. The following image displays.



2. Click the Design Table icon ().
3. Click the **Create a design table from a pre-existing file** radio button and click **OK**.
4. In the opening **File Selection** window, select the KwrAddARow.xls file and click **Open**.
5. Click **Yes** in the Automatic associations window: The design table opens. Click **OK** to close it.
6. Click the **Measure update** icon to update Formula.1.
7. Under the Design Tables node, double-click **Configuration=1**. The **Edit Parameter** dialog box displays.
8. Click the Design table icon in the **Edit Parameter** dialog box: The Design Table window displays.

9. In the dialog box, select the second configuration (line 2), click **Apply**, and **OK** twice.

10. Right-click Formula.1 in the specification tree and select the **Local Update** command.

11. In the Specification tree, right-click DesignTable.1 and select the **DesignTable.1 object->Deactivate** command. The design table is deactivated.

12. Modify the spline, to do so, proceed as follows:

<table>
<tr><td colspan="2">

- Double-click Point.1 twice in the specification tree or in the Geometry. Enter the coordinates indicated below into the Point Definition dialog box.

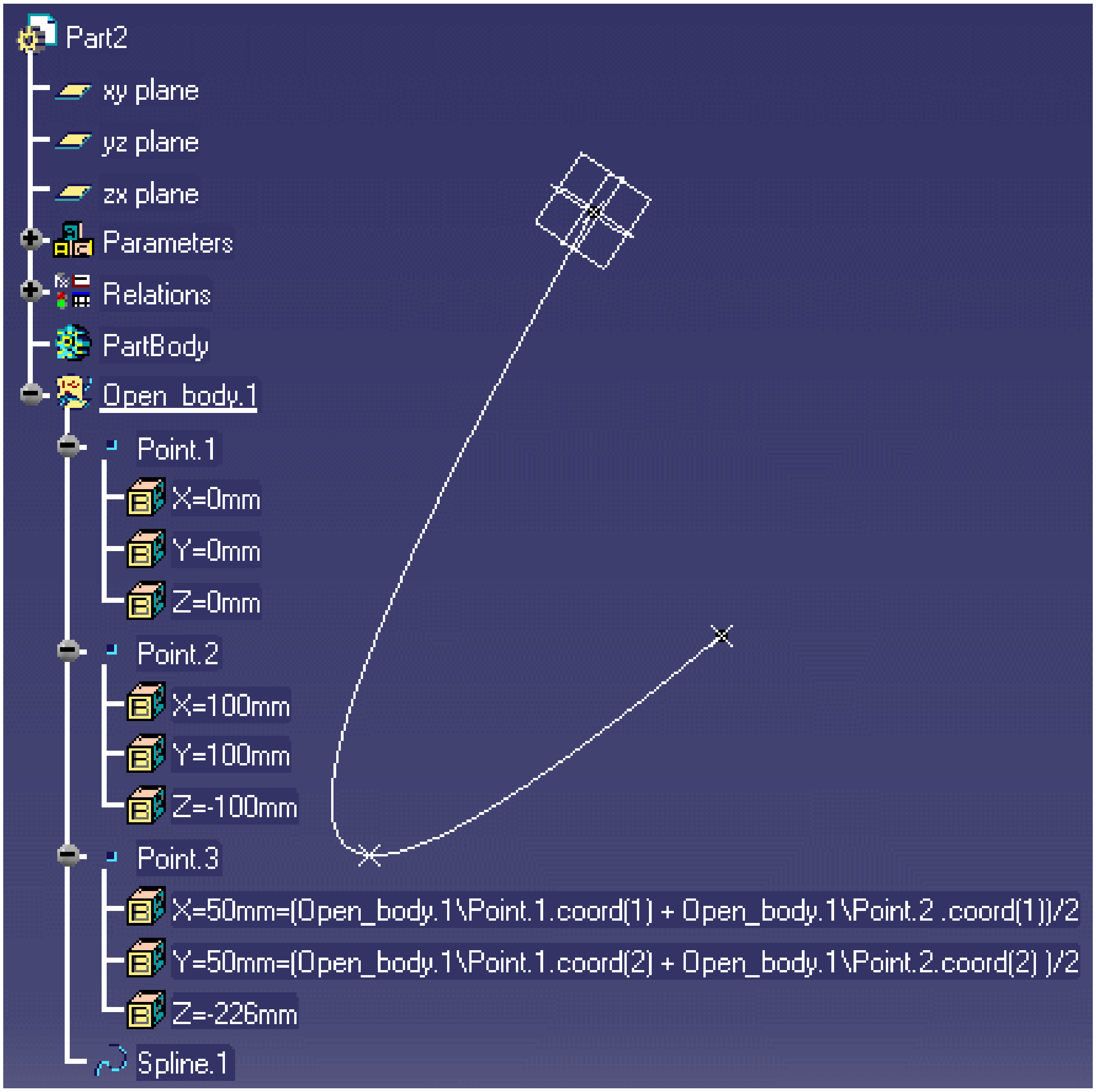
</td></tr>
<tr><td>

- Modify the coordinates of Point.2 and Point.3 (see table opposite)

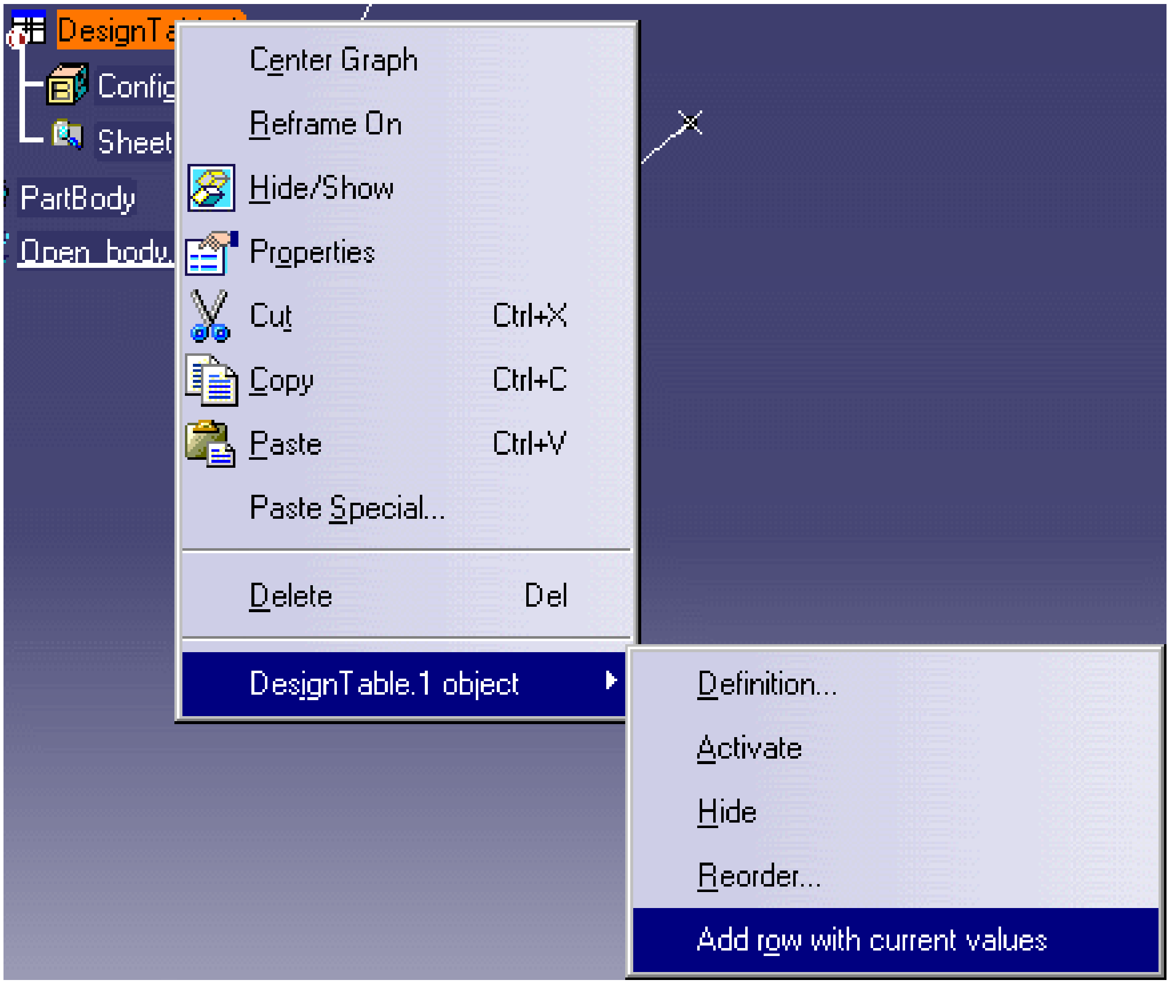
</td><td>
<table>
<tr><th></th><th>Point 1</th><th>Point 2</th><th>Point 3</th></tr>
<tr><td>X</td><td>0</td><td>100</td><td>50</td></tr>
<tr><td>Y</td><td>0</td><td>100</td><td>50</td></tr>
<tr><td>Z</td><td>0</td><td>-100</td><td>-226</td></tr>
</table>
</td></tr>
<tr><td colspan="2">

- Click **OK** when done.

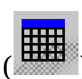
</td></tr>
</table>



13. Add the new configuration to the design table. To do so, right-click DesignTable.1 in the specification tree and select the **DesignTable.1 object->Add row with current values** command.



14. Right-click DesignTable.1 and select the **DesignTable.1 object->Activate** command.

15. Double-click **Configuration=1** under DesignTable.1 and click the Design Table icon ().

16. In the DesignTable.1 window select the configuration that you have just added and click **Apply** and **OK** twice. The spline is updated accordingly.

Controlling Design Tables Synchronization

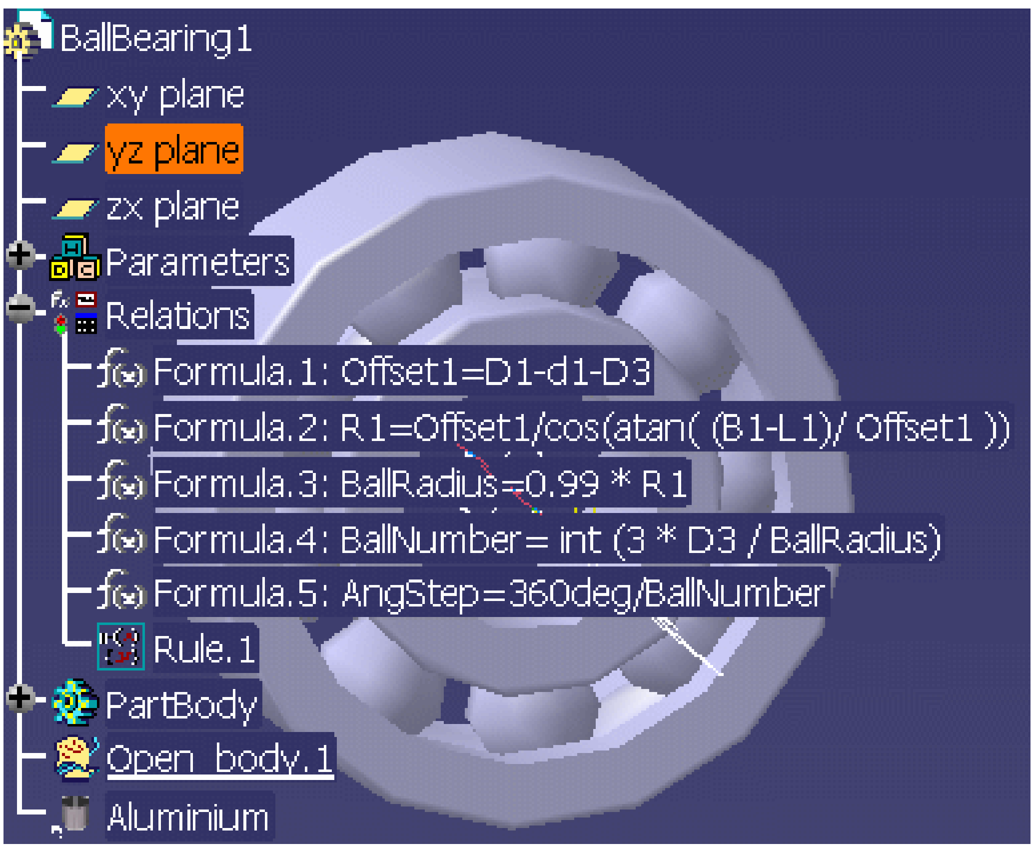
This topic aims at providing the user with short examples when working with design tables in the following modes:

- Automatic Synchronization At Load
- Interactive Synchronization At Load
- Manual Synchronization

Automatic Synchronization At Load

When loading a model containing user design tables, if the design table files have been modified and the external file data is contained in the model, the design table will be synchronized automatically if this radio button is checked.

1. Open the KwrBallBearing1.CATPart file. The following image displays.



2. Click the **Design Table** icon ( ). The Creation of a Design Table dialog box displays.

3. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.

4. Select the KwrBearingDesignTable.xls file and click **Open**. Click **Yes** when asked if you want to associate the columns of the tables with the parameters.

5. Click **OK** to apply the default configuration.

6. Save your file and close it.

7. Open the KwrBearingDesignTable.xls file. Change the material of row 2 to Gold. Save your file and close it.

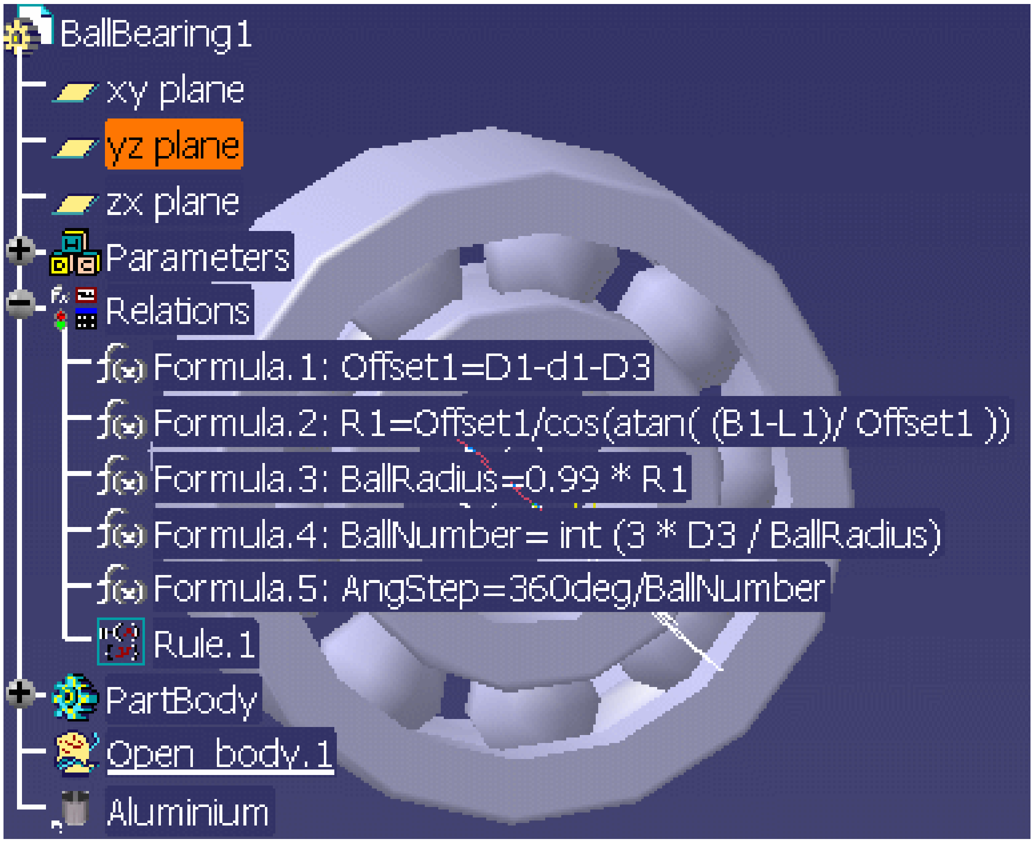
8. Go back to Catia. Open the part: The Part is updated accordingly to your changes.

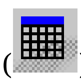
Interactive Synchronization At Load

When loading a model containing user design tables whose external source file was deleted, this option enables the user to select a new source file or to save the data contained in the design tables in a new file.

1. From the **Tools->Options...** menu, select **General->Parameters and Measure** and check the **Interactive Synchronization At Load** option in the Knowledge tab.

2. Open the KwrBallBearing1.CATPart file. The following image displays.



3. Click the Design Table icon (). The Creation of a Design Table dialog box displays.

4. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.

5. Select the KwrBearingDesignTable.xls file and click **Open**. Click **Yes** when asked if you want to associate the columns of the tables with the parameters.

6. Click **OK** to apply the default configuration.

7. Save your file and close it.

8. Go to the directory containing the KwrBearingDesignTable.xls file and delete it.

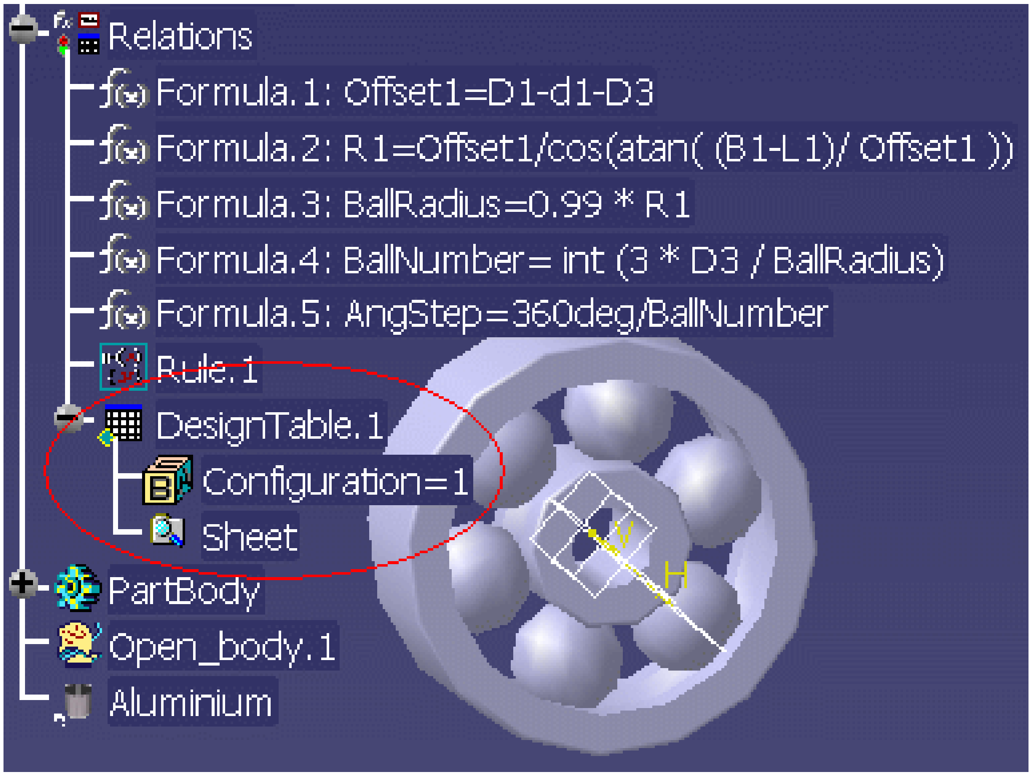
9. Go back to Catia. Open the KwrBallBearing1.CATPart file: A dialog box displays asking you if you want to select a new file. Click the **Select** button and select a new Excel file.

Manual Synchronization

When loading a model containing user design tables, if the design table files have been modified and the external file data is contained in the model, the design table will be synchronized if this option is checked. To synchronize both files, right-click the design table in the specification tree and select the **DesignTable object->Synchronize** command or the **Edit->Links** command.

1. From the **Tools->Options...** menu, select **General->Parameters and Measure** and check the **Manual Synchronization At Load** in the Knowledge tab.

2. Open the KwrBallBearing2.CATPart file. This file already contains a design table whose values are identical to those contained in the KwrBearingDesignTable.xls file (Note that the KwrBearingDesignTable.xls file and the KwrBallBearing2.CATPart file should be located in the same directory.)



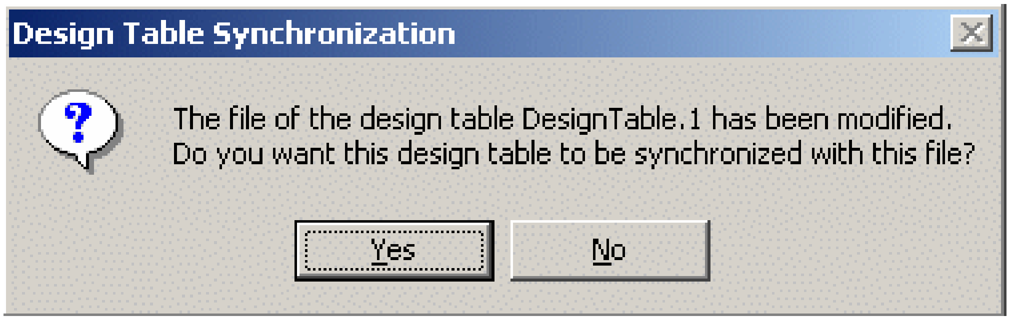
3. Select the **Edit->Links** command to edit the Excel file path and select the appropriate KwrBearingDesignTable.xls file. Save the file and close it.

4. Open the KwrBearingDesignTable.xls file and modify the material values for example. Close the file.

5. Go back to *CATIA*. Open the KwrBallBearing2.CATPart file.

6. Select the **Edit->Links** command and click the Synchronize button to synchronize both files.

If the **Duplicate data in *CATIA* model** option is checked, and if you choose another design table file without using the **Edit Table** command when in session, the following message displays whatever the settings:



If the **Duplicate data in *CATIA* model** option is unchecked, the synchronization occurs automatically.

Storing a Design Table in a PowerCopy

This task shows how to store a design table in a power copy for later use. In this scenario, the user wants to instantiate the inner and the outer cages of a ball bearing in a different context. To do so, he creates a powercopy only containing the outer and the inner cages of an already existing ball bearing.

This scenario is divided into the following steps:

- Inserting the Design Table into the CATPart file
- Creating the PowerCopy
- Instantiating the PowerCopy containing the Design Table

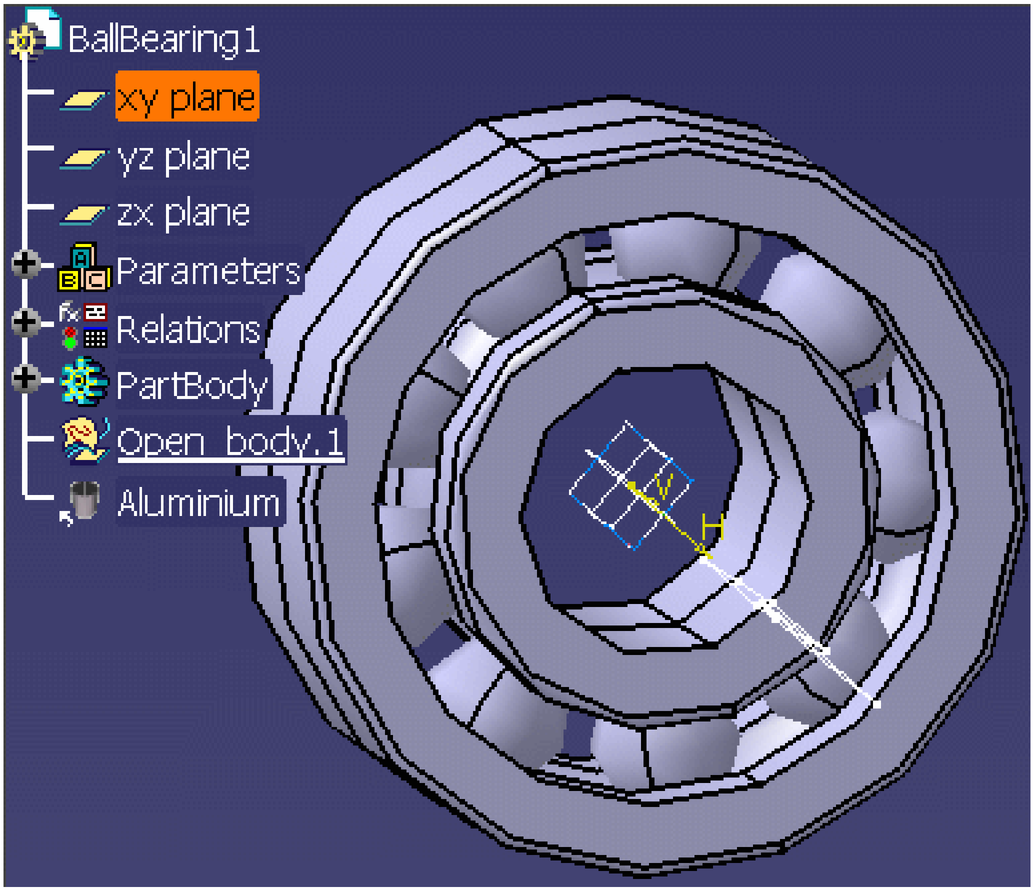
To carry out this scenario, the Product Knowledge Template license is required.

To carry out this scenario, you will need the following files:

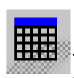
- KwrBallBearing1.CATPart
- KwrBearingDesignTable.xls

To store a design table in a PowerCopy, do not forget to select the parameters pointed by the design table.

1. Open the KwrBallBearing1.CATPart file. The following image displays.



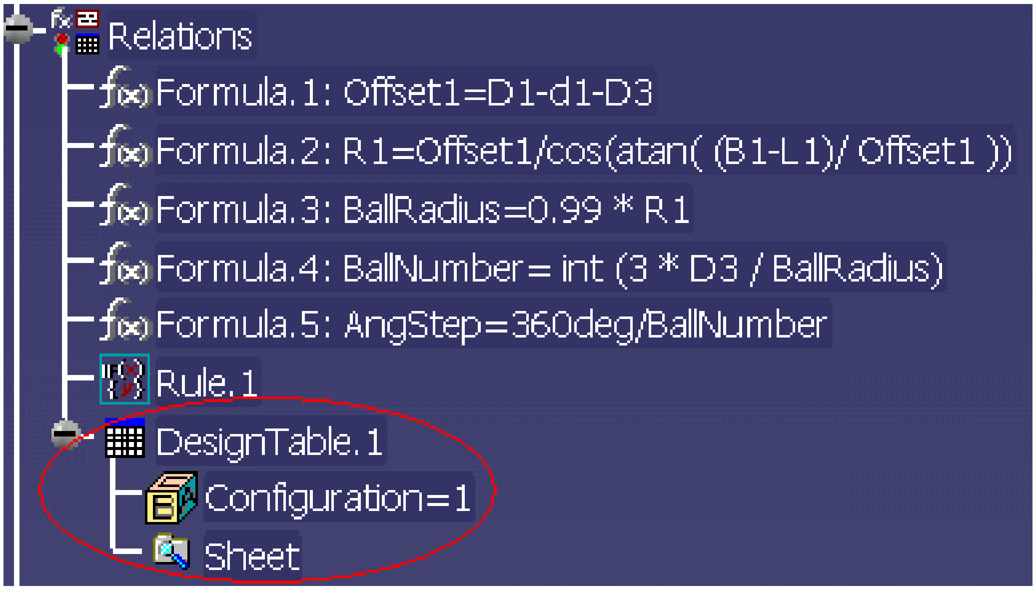
Inserting the Design Table into the CATPart file

2. Click the **Design Table** icon () in the Standard toolbar. The **Creation of a Design Table** dialog box displays.

3. Check the **Create a design table from a pre-existing file** radio button and click **OK**. The **File Selection** dialog box displays.

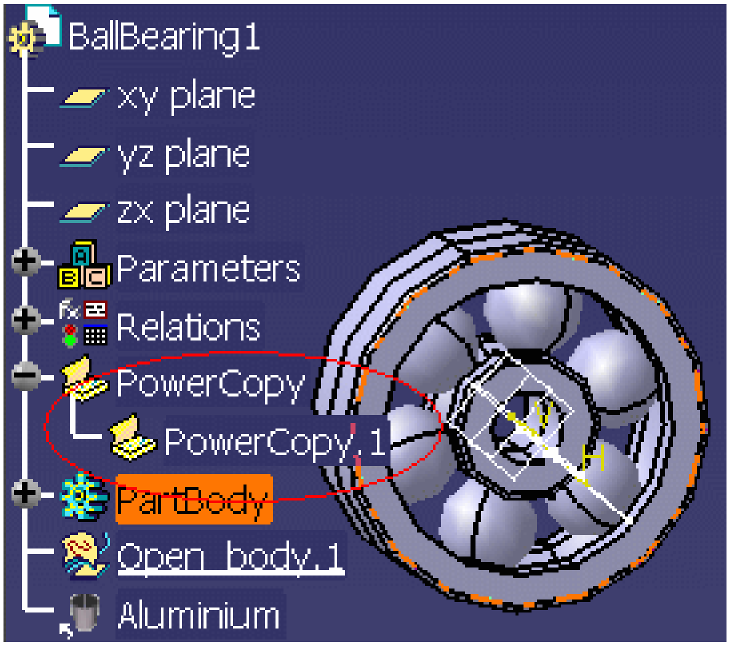
4. Select the KwrBearingDesignTable.xls and click **Open**.

5. Click **Yes** when asked for automatic associations and click **OK**. The Design table now displays below the Relations node.



Creating the PowerCopy

6. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench (if need be) and click the **Create a PowerCopy** icon. The Powercopy Definition dialog box displays.
7. In the Specification tree, select the following items:
   - DesignTable.1
   - Shaft.1
   - Shaft.2
   - Shaft.3
   - Sketch.1
   - Sketch.2
   - Sketch.3
   - the Material Parameter.
   - Click **OK** when done. The PowerCopy displays below the PowerCopy node in the specification tree



8. Save your file and close it.

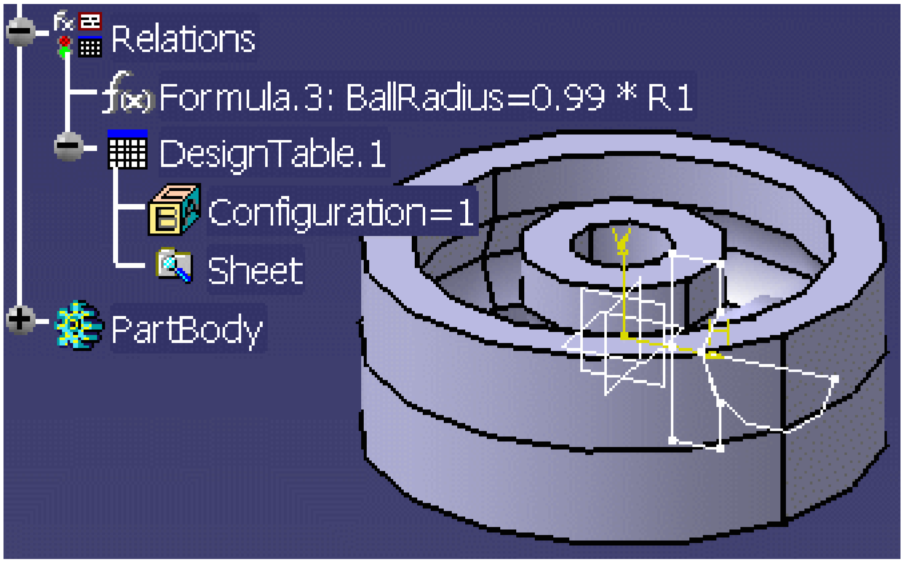
Instantiating the PowerCopy

9. From the **File->New** menu, select **Part** in the **List of Types** and click **OK**.

10. If need be, from the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench and click the **Instantiate From Document** icon. The **File Selection** dialog box displays.

11. Select the KwrBallBearing1.CATPart file and click **Open**. The **Insert Object** dialog box displays.

12. Select the yz plane in the specification tree and click **OK**. The Design Table is instantiated


Relations
Formula.3: BallRadius=0.99 * R1
DesignTable.1
Configuration=1
Sheet
PartBody

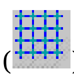
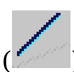
Creating and Using a Knowledge Advisor Law

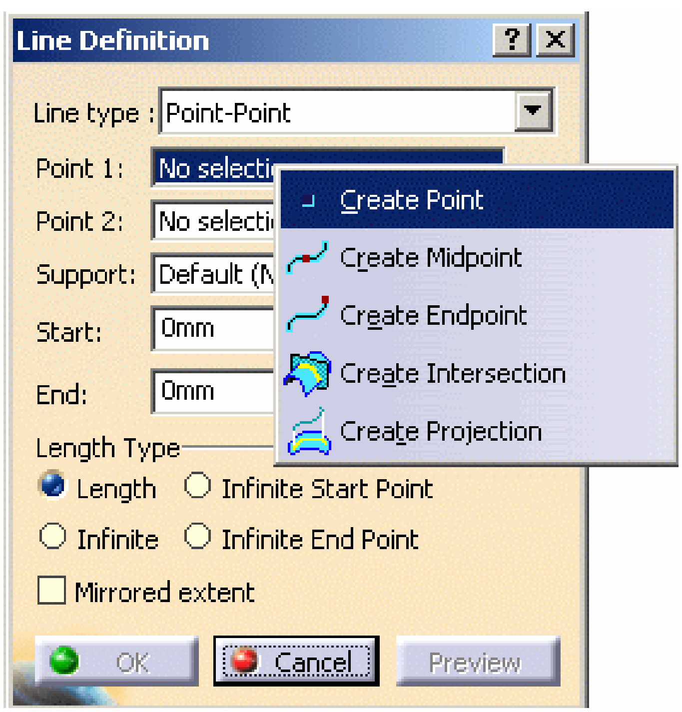
The scenario which is developed below illustrates how to create a Knowledge Advisor law, then use a combination of a Generative Shape Design law and a Knowledge Advisor law in the same relation.

- The Evaluate method is to be used to calculate a parameter value when this parameter is defined by a Generative Shape Design law.
- Note that the result you obtain on completion of this task depends on the initial lines. You can replay the scenario with different lines and see how it affects the result.

A Knowledge Advisor law is a relation whereby a parameter is defined with respect to another. Both parameters involved in a law are called *formal parameters*. Formal parameters and laws are specifically designed to be used in the creation of shape design parallel curves. A Generative Shape Design law can be used in a Knowledge Advisor law.

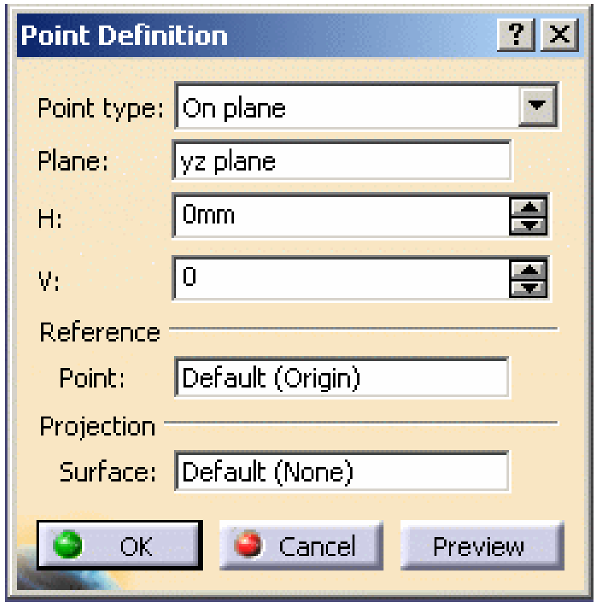
Laws only specify a relation between one parameter and another single parameter.

1. From the **Start->Shape** menu, access the **Generative Shape Design** workbench.
2. Define a working support using the **Work on Support** icon ().
3. Select the zx plane, for example, and click **OK** in the updated Work on Support dialog box without modifying any other parameter.
4. Click the Line icon (). The Line dialog box is displayed.
5. Right-click in the Point 1 field, and choose the **Create point** command.



6. The Point Definition dialog box is displayed, the **Point type** and **Plane** fields being automatically filled.

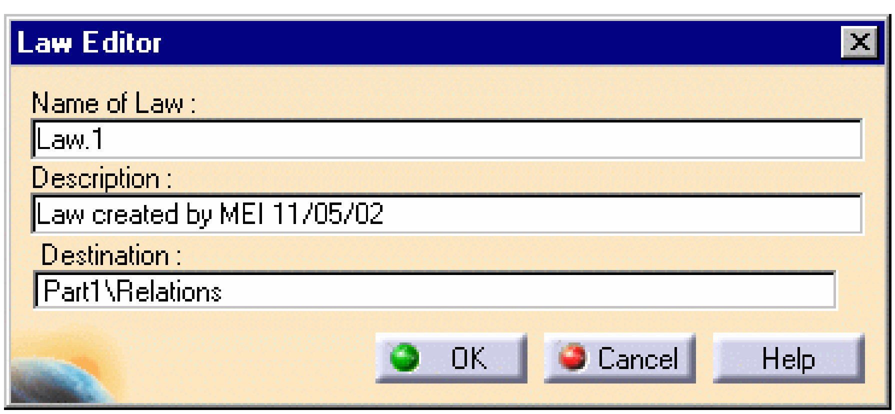
7. Create a point at **H:0mm** and **V:0mm**, and click **OK**.



8. Repeat the operation, right-click the Point 2 field from the Line dialog box to create another point at **H:100mm** and **V:0mm**, then click **OK** in the Point Definition dialog box.

9. Click **OK** in the Line dialog box to create the line.

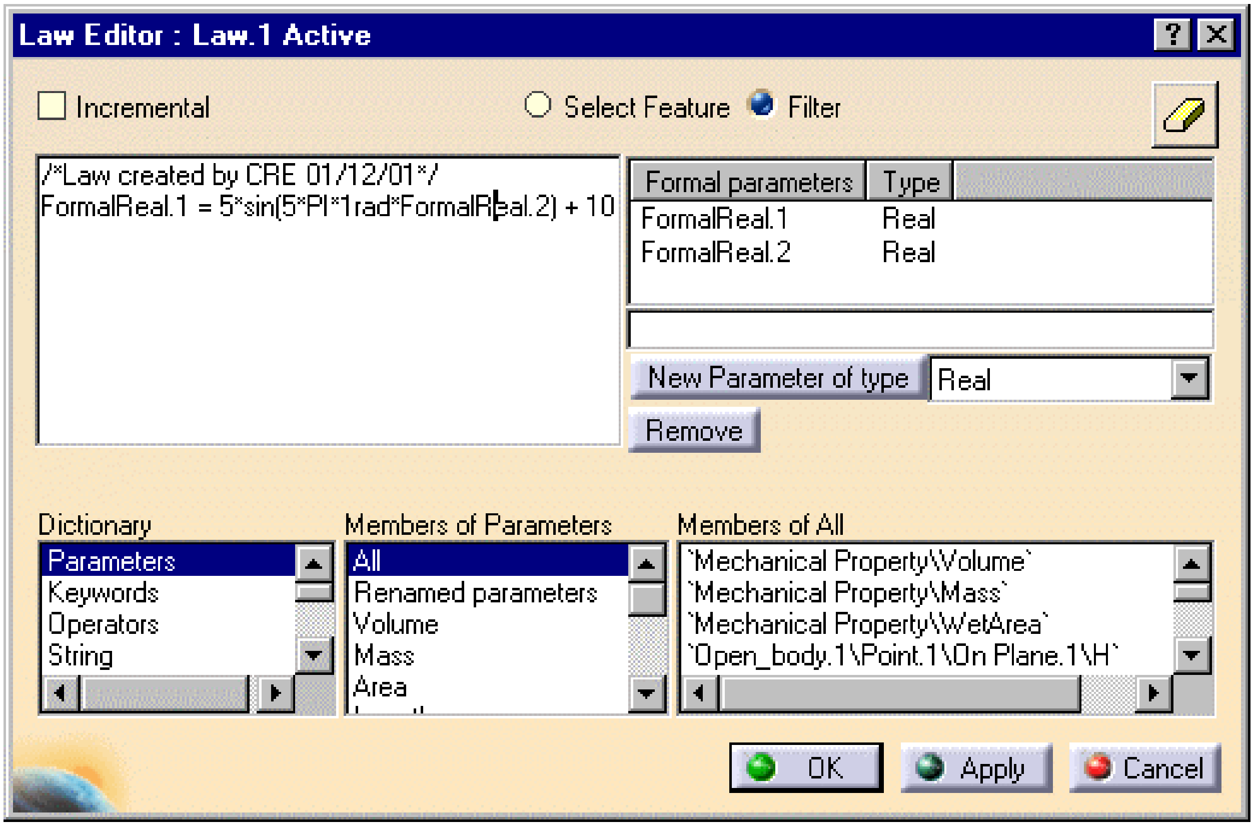
10. Access the Knowledge Advisor workbench and click the  icon. If need be, use the **Tools->Customize** command to access the icon. A dialog box similar to the one below is displayed. This editor is similar to the other relation editors. If need be, replace the default values specified in the dialog box fields.



11. Click **OK**. The law editor is displayed. The right-hand part allows you to create the parameters to be used in the law. The left-hand part is the law edition box.

12. Click the **New Parameter of type** button to create two real type parameters FormalReal.1 and FormalReal.2, then enter the law below into the edition window:

FormalReal.1 = 5*sin(5*PI*1rad*FormalReal.2)+ 10

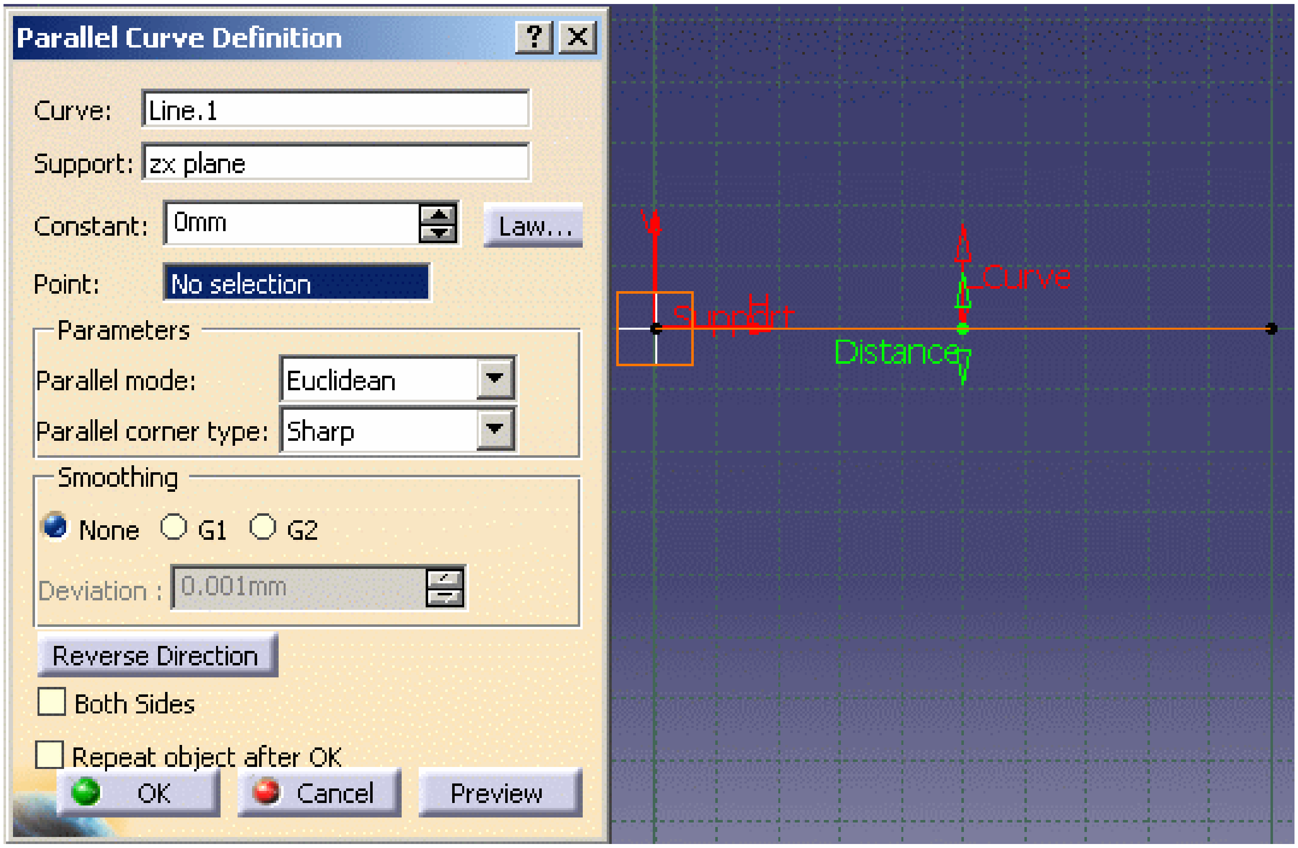


13. Click **OK** to add the law to the document. The Law.1 feature is added to the specification tree right below the Relations node.

14. Select your document root feature and re-access the Generative Shape Design workbench.

15. Click the  icon to create a curve parallel to the line created at the very beginning of the scenario. The **Parallel Curve Definition** dialog box is displayed.

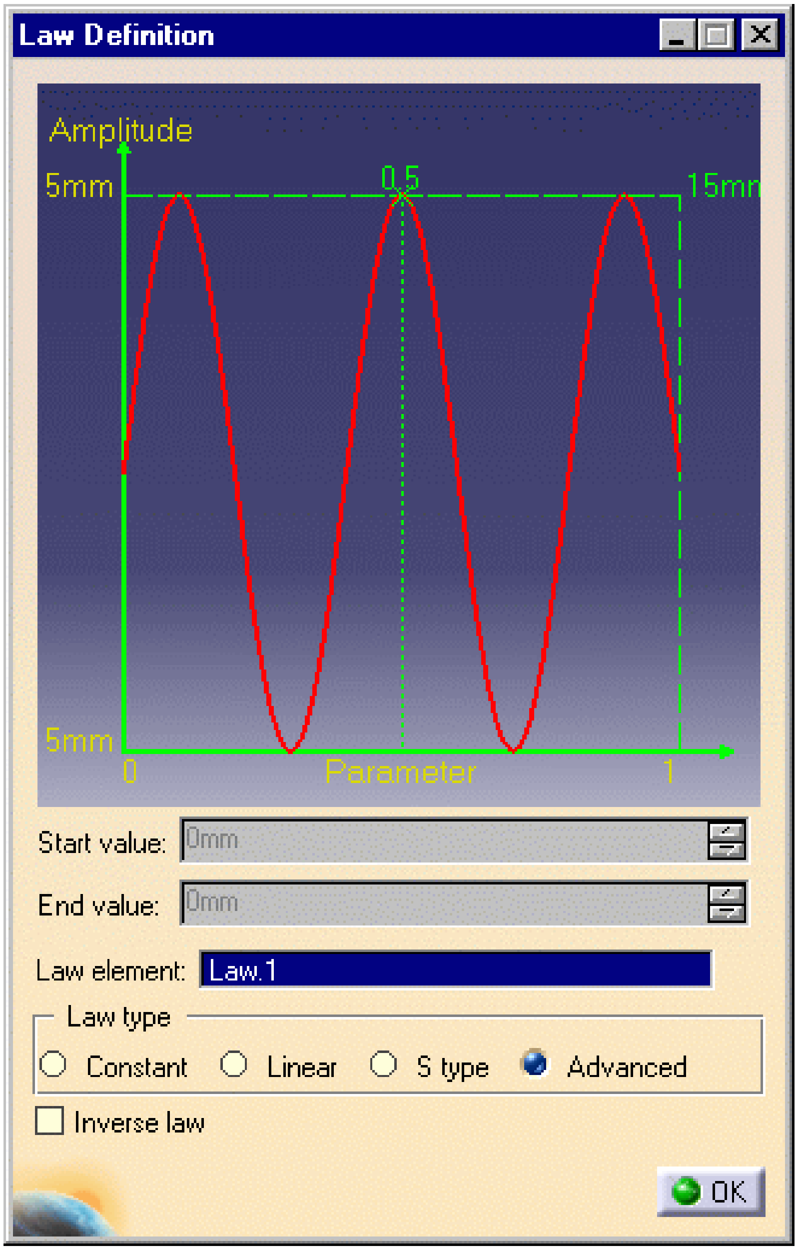
16. Select the line that you previously created as the reference **Curve**.



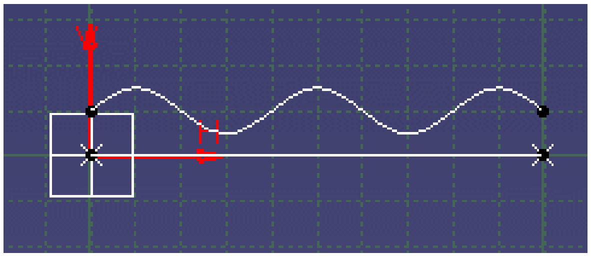
Note that only positive laws, i.e. with positive values only, can be used when creating parallel curves (positive is to be understood as "strictly positive").

18. Click the **Law ...** button. The Law Definition dialog box displays.

19. Click the **Advanced** Law type, click Law.1 in the specification tree and click **Close**.



20. Click **OK**. A curve parallel to the selected one is created, taking the law into account.



The KwrCreatingaLaw.CATPart sample illustrates this scenario.

Using the Knowledge Inspector

The Knowledge Inspector allows you to query a design to determine and preview the results of changing any parameters without committing themselves to actually changing the design. This "what if" analysis provides immediate feedback that helps you experiment and refine designs.

While it is important to determine what happens when one or more parameters are changed, it is equally significant for you to see how a design can be changed to achieve a desired result. The Knowledge Inspector supports this by allowing you to query "how to" make a particular change.

In short, the Knowledge Inspector is a tool designed to study impacts and dependencies.

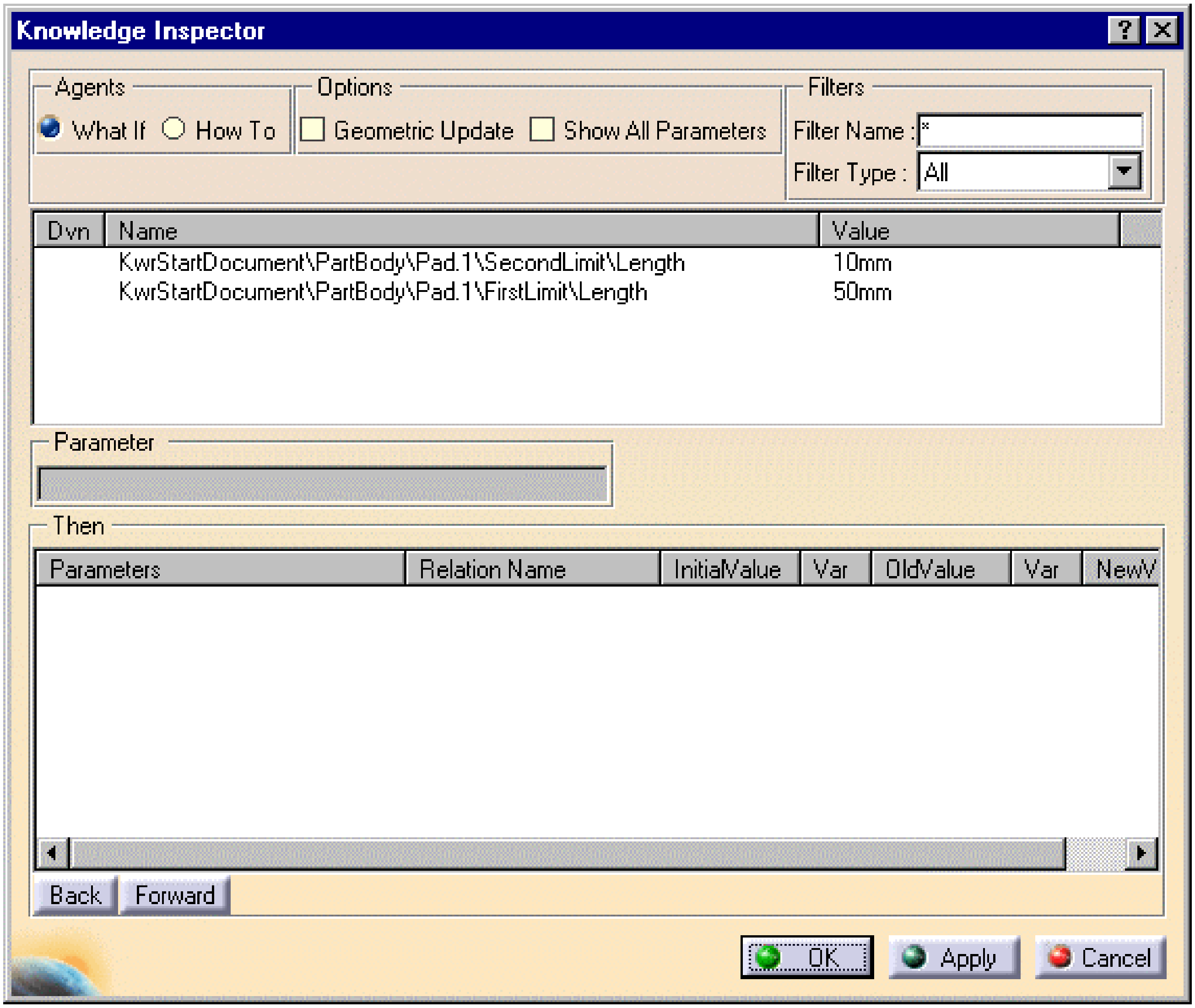
What if (*impacts*)	Helps you understand to what extent changing any parameter of your design (such as material, pressure, or a dimensional parameter) changes the operation or design of the product on which you are working. Can be used to examine interactions of parameters with each other and with the rules that make up the product's specifications. A "Geometric Update" option enables you to visualize the result of your modification in the geometry area.
How To *(dependencies)*	Helps you determine how your design can be changed to achieve a desired result.

You shouldn't use the f(x) capabilities with the Knowledge Inspector.

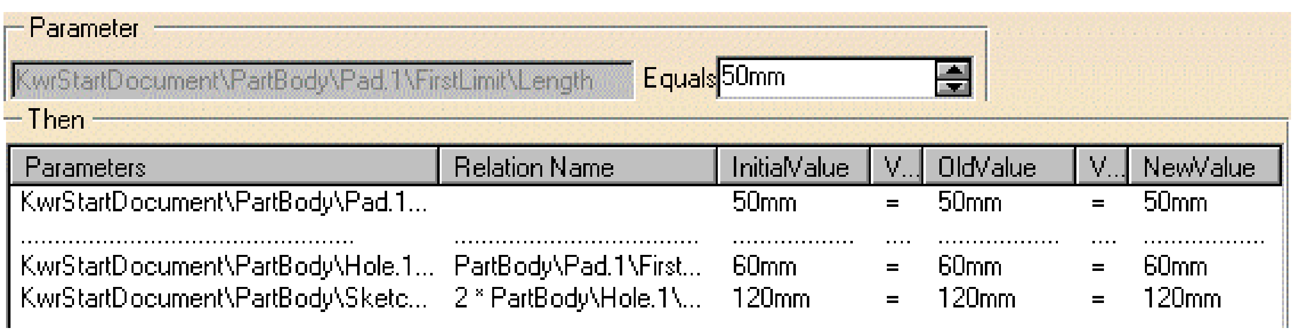
The 'What If' Mode

This task explains how to use the 'What If' mode.

1. Open the KwrFormula1.CATPart document and access the Knowledge Advisor workbench.
2. Click the Knowledge Inspector icon or select the Knowledge Inspector from the standard tool bar. The "Knowledge Inspector" dialog box is displayed. Check the 'What If' option.
3. Select the KwrStartDocument\PartBody\Pad.1\FirstLimit\Length parameter (at this stage, don't modify its value in the Equals field).

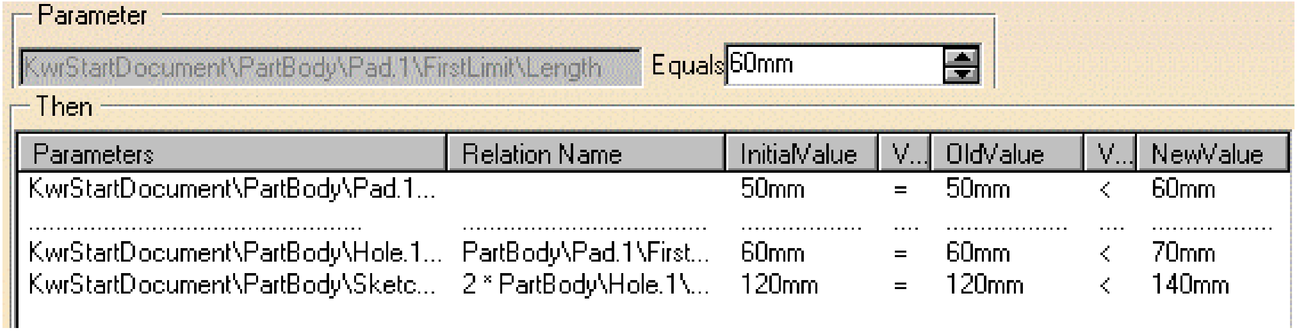


4. Click **Apply**. The following list of parameters and parameter values is displayed in the Then area.



The first line describes the parameter which has just been selected. The other lines describe the impacted parameters.

5. Use the Equals field to replace the KwrStartDocument\PartBody\Pad.1\FirstLimit\Length parameter value with 60mm. Click **Apply**. In the Then area, the parameter values are updated as follows:



Parameters	Relation Name	InitialValue	V...	OldValue	V...	NewValue
KwrStartDocument\PartBody\Pad.1...		50mm	=	50mm	<	60mm
....	....	....	....	....	....	....
KwrStartDocument\PartBody\Hole.1...	PartBody\Pad.1\First...	60mm	=	60mm	<	70mm
KwrStartDocument\PartBody\Sketc...	2 * PartBody\Hole.1\...	120mm	=	120mm	<	140mm

The InitialValue column shows the initial parameter values (when you open the Knowledge Inspector). The OldValue column shows the parameter values resulting from the previous 'What if' operation. The Var (variations) columns show comparison operators between values located in adjacent columns.

6. Check the **Geometric Update** option to display in the geometry area the modifications resulting from the 'What If' operation. Click **Apply** to update the document in the geometry area.

7. Click **OK** to apply the values resulting from the current 'What If' operation to your document. Otherwise, click Cancel>.

Note that:

- Using the 

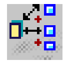
buttons reloads in the 'Then' area the previous or next values in the history of the 'What if" operations.
- Checking the **Show All Parameters** option displays all the document parameters. An `f` letter in the Dvn column indicates that the parameter is constrained by a formula.
- Selecting a parameter in the **Then** area while the **Show All Parameters** is checked, highlights the selected parameter in the parameter list above.

Modifying a parameter value does not imply that the values of the impacted parameters are automatically updated by a 'What If' operation. For example, if a parameter is constrained by a formula such as:
if Parameter1 > A then Parameter2 = B
replacing the *Parameter1* value with a value greater than A won't modify Parameter2 if Parameter2 was previously set to B.

The 'How To' Mode

This task explains how to use the 'How To' mode.

1. Open the KwrFormula1.CATPart document.

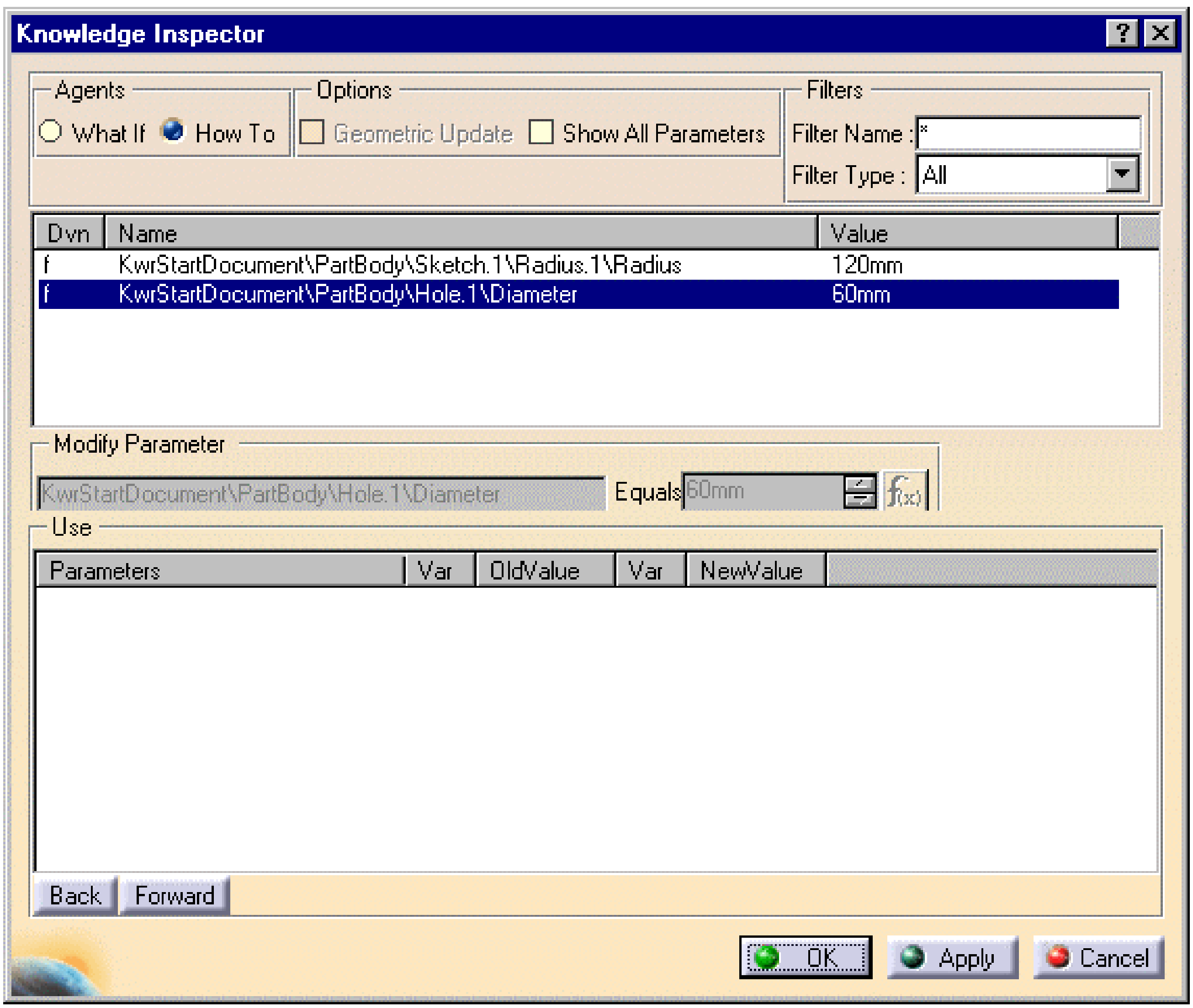
2. Click the Knowledge Inspector icon  in the standard toolbar. The Knowledge Inspector dialog box is displayed. Check the '**How To**' option. By default, only the parameters which are constrained by a formula are displayed.

3. If need be, check the Show **All Parameters** to display all the document parameters.

4. Select the KwrStartDocument\PartBody\Hole.1\Diameter parameter (assuming that you would like to have this parameter modified).

5. Click **Apply** or **Enter**. The list of parameters to be modified in order to change the Hole.1\Diameter parameter is displayed in the 'Use' area.

6. Select the Pad.1\FirstLimit\Length parameter.



7. Check the **What If** option.

8. Modify the FirstLimit\Length parameter in 'What If' mode.

9. Click OK to apply the parameter modification to your document.

Note that:

- Checking the Show All Parameters option displays all the document parameters. An f letter in the Dvn column indicates that the parameter is constrained by a formula.
- Selecting a parameter in the 'Use' area while the **Show All Parameters** is checked, highlights the selected parameter in the parameter list above.

Working with the Rule Feature

Select the Rule icon to create a rule, write its code, test its syntax and apply it to your document.

A rule is a set of instructions, generally based on conditional statements, whereby the relationship between parameters is controlled. In addition, depending on the context described by the rule instructions, actions can be executed:

- To set a value or a formula to parameters, including feature activity
- To display information panels
- To launch Visual Basic macros stored in external files or in the V5 document.
- To affect points, curves and surfaces and thus allow contextual and automatic topological changes

In the specification tree, the rule is displayed as a relation that can be activated or deactivated. Like any feature, a rule can be manipulated from its contextual menu.

A rule is executed when one of its input parameters has changed or when one of its input features has changed and if the user requires the update of the rule.

The consequence is that it is impossible for the user to completely control when the rule is to be fired. As a result, rules should only manipulate parameters and features and should be used to control the status of a design (change of parameters and geometry).

If the user wants to control when the action takes place, he should use the Reaction feature.

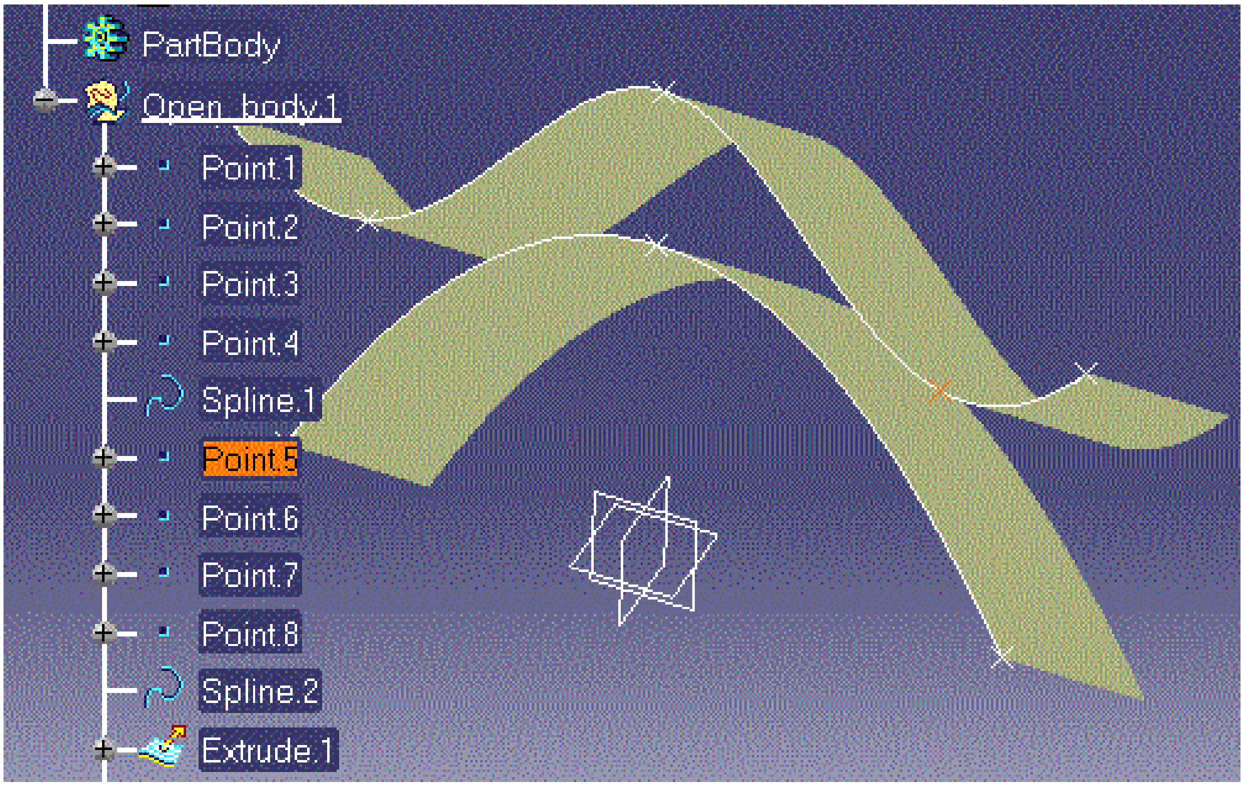
Creating a Rule
Using the Rule Editor
Using Rules and Checks in a Power Copy
Using the Dictionary

Creating a Rule

The task described below explains how to create a rule which retrieves the abscissa of a point and, depending on the coordinate value, displays a message or another.

This scenario uses two special functions allowing you to retrieve the coordinates of a point. These functions can be accessed from the Measures item of the Dictionary.
To know more about the Dictionary, see the Using the Dictionary.

1. Open the KwrMeasure.CATPart document. The whole document has been created using the Generative Shape Design product. The extruded surfaces are extruded from the Spline.1 and Spline.2 curves. The point whose coordinates are to be retrieved and tested is Point.5.



2. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench.

3. Use the editor to create three Length type parameters: Point5X, Point5Y and Point5Z.

4. Click the icon. In the first dialog box which is displayed, enter a rule name (MeasureRule for example). If need be, replace the default comments. If you want to add the rule to be created to a specific relation set, specify a destination. To do so, see Creating Sets of Relations.

5. Click OK. The Rule Editor is displayed.

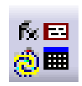
6. Enter the rule below in the edition window.

```
if Geometrical Set.1\Point.5.coord(1) > 0mm
Message("Point.5 abscissa is positive")
else
{
  Geometrical Set.1\Point.5.coord(Point5X, Point5Y, Point5Z)
  Message("Point.5 abscissa is:  # ", Point5X)
}
```

7. In the rule above, you can retrieve the Point.5 definition (Geometrical Set.1\Point.5) by double-clicking the feature in the specification tree.

8. Click **OK**. The message "Point.5 abscissa is: 0mm" is displayed.

9. Edit the Point.5 feature (double-click the object in the specification tree for example) and replace the Point.5 X value with 10 mm. The rule is in a to-be-updated status. See Updating Measures for information on relations to be updated.

10. Re-access the Knowledge Advisor workbench, then click the  icon. A message box informs you that "Point.5 abscissa is positive").

To know more about the Rule Editor, see Using the Rule Editor.

Using Rules and Checks in a Power Copy

This task explains how to use rules and checks in a Power Copy.

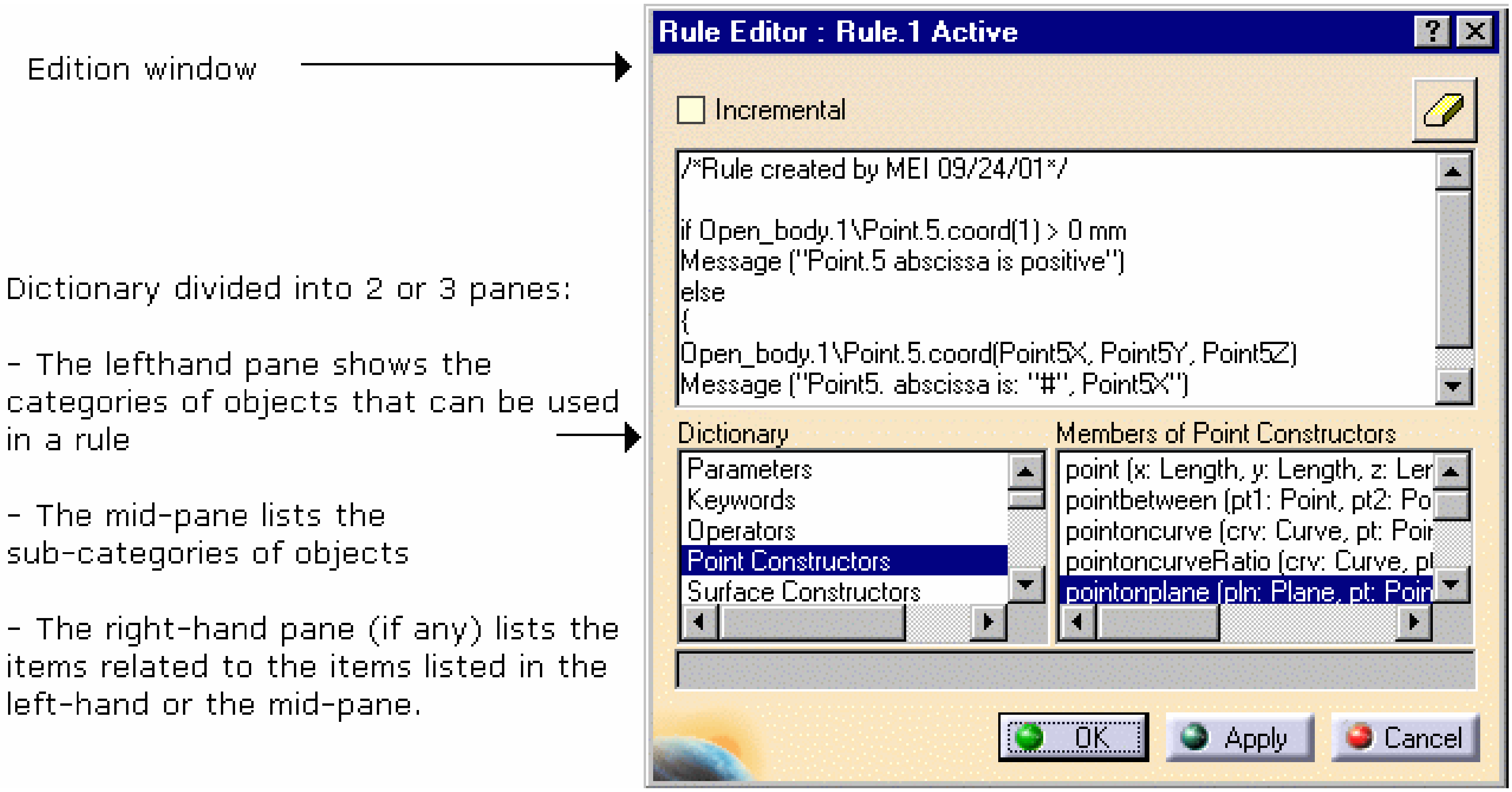
Rules and checks as well as other relations can be applied to a document by retrieving them from another document provided they have been stored in a *power copy*. For further information on the power copy mechanism, see the *Generative Shape Design User's Guide*.

1. Open the KwrMeasurePCopy.CATPart document. If need be, access the Generative Shape Design workbench.
2. In the standard menu bar, select the **Insert->Knowledge Templates->PowerCopy...** command. The Power Copy definition window is displayed.
3. In the specification tree, select the Rule.1 and Check.1 relations. Both relations are carried forward onto the Power Copy definition panel. Click **OK** in the Power Copy creation panel. Save and close your document.
4. Open the KwrSplineInPcopy1.CATPart document and access the Generative Shape Design workbench.
5. Select the **Insert->Instantiate From Document...** command from the standard menu bar. The Select PowerCopy dialog box is displayed. Select the document which contains the power copy storing the Rule.1 and Check.1 relations and click **Open**. The **Insert Object** dialog box is displayed.
6. Select the Spline.1 feature either in the specification tree or in the geometry area. Click **OK**. A message box launched by the check is displayed informing you that the Spline Length is < 100mm. Both relations are carried forward to the specification tree and the check icon is red. The rule has not been fired.
7. Open the KwrSplineInPcopy2.CATPart document and repeat the same operation (from step 5). An information box displays the Spline Length indicating that Rule.1 is fired. This time, the check icon is green is the specification tree.

Rules and checks can be stored in catalogs and instantiated later in a document. See Instantiating Knowledgeware Relations from a Catalog

Using the Rule Editor

The Rule Editor is intended to help the user key in the check body using the Dictionary. It is made up of:



In the Rule Editor, you can:

- Restrict the list of parameters displayed in the dictionary: in the specification tree, simple click the feature you want to display the parameters. If the 'Incremental' option is selected, only the first level of parameters right below the selected feature are displayed, otherwise, all the parameters at all levels are displayed. Suppose your document contains a Geometrical Set feature which itself is made up of several Shape Design points. When the 'Incremental' box is unchecked, selecting the Geometrical Set feature in the specification will display all the parameters related to the points (the parameters which defines the coordinates are included in the list). When the 'Incremental' box is checked, selecting the Geometrical Set feature displays only the first level of parameters below the Geometrical Set (the point coordinates are not displayed).
- Insert the feature definition in a rule: in the specification tree, double click the feature you want to insert the definition.
- Check whether the rule syntax is correct: click **Apply**.
- Erase the contents of the edition window: click the  icon.
- Add the rule to the document: click **OK**.

To know more about the items displayed in the Dictionary, see Using the Dictionary or select one of the items and press the F1 key in Catia.
To know more about rules, see Creating a Rule.

Handling Errors

This task explains how to handle features that are in error. In the scenario described below, the user opens a .CATPart file that is made up of 2 lines an a datum point. He wants this point to be the intersection of the lines if they intersect, or the origin if they do not. To do so, he creates a rule.

The use of geometrical operators to value geometry in relations may lead to update errors in the created features. If the user values a datum point with the result of the intersection of 2 lines, these 2 lines may not intersect and the point is therefore in error.

The **Do not catch evaluation errors** option which is available by selecting the relation in the specification tree and selecting the Properties command enables the user:

- To create features in error
- To know (through an error message) if a feature is in error and
- To make changes (or not) in case of errors.

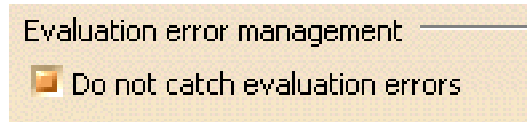
Note that this error handling is only available for GSD features.

1. Open the KwrErrorHandling.CATPart file.
2. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench.
3. Click the Rule icon ( ) to create a rule that will react to the fact that the datum point is the intersection of the 2 lines if they intersect or the origin if they do not.
4. In the Rule Editor, enter the following body:

```
let x(Point)
x =intersect(`Geometrical Set.1\Line.1` ,`Geometrical Set.1\Line.2` )
if (x.Error==true)
`Geometrical Set.1\Point.5` = point(0mm,0mm,0mm)
else
`Geometrical Set.1\Point.5` = x
```

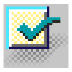
5. Click **OK** when done. The intersect operator launches an error message if the intersection is empty.

6. Right-click the rule in the specification tree and select the **Properties** command.



7. Check the **Do not catch evaluation errors** check box and click **OK** when done. This way, the evaluation of the rule will be forced even if an update error occurs.

Working with the Check Feature



Select the Check icon to create a check, write its code, tests its syntax and apply it to your document.

A check is a set of statements intended to inform the user if certain conditions are fulfilled or not. A check does not modify the document it is applied to. A check is a feature. In the document specification tree, it is displayed as a relation that can be activated and deactivated. Like any feature, a check can be manipulated from its contextual menu.

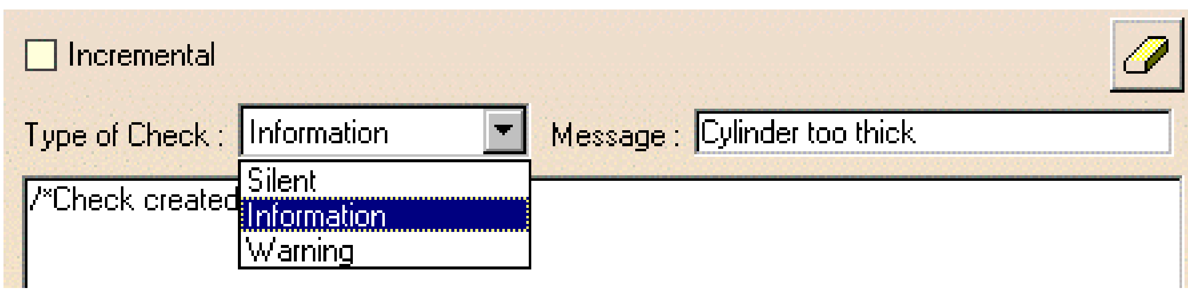
Creating a Check
Using the Check Editor
Performing a Global Analysis of Checks
Using the Check Analysis Tool
Using Rules and Checks in a Power Copy
Customizing Check Reports
Using the Dictionary

Creating a Check

This task explains how to create a check.

1. Open the KwrFormula0.CATPart document, select the root item in the specification tree and access the Knowledge Advisor workbench.
2. Click the Check icon . The first Check Editor dialog box is displayed.
3. Replace the default name with Cylinder_Check. If needed, add some comments to the Description field.

   If you want to add the check to be created to a specific relation set, specify a destination. To do so, see Creating Sets of Relations. By default, the check is created right below the Relations node.
4. Click OK. The Check Editor is displayed.



5. Select a type of check. Enter the message you want to be displayed in the information or warning box in case the check is not verified.
6. Enter the check statements in the edition window. You can simply Copy/Paste the following statements into the edition window:

   Relations\Formula.1\Activity == false
7. Click **Apply** to test your check syntax. If the information message displays, the check syntax is correct.
8. Click **OK** to add Cylinder_Check to the relations node in the specification tree. A red icon is displayed in the specification tree meaning that the check is not valid.

9. Deactivate Formula.1, the check icon turns to green in the specification tree.

Three parameters related to a check are displayed in the "Formulas" dialog box:

- The activity
- The severity
- The result

When you select the result parameter, the icon indicating whether the check is valid or not is displayed opposite the value field. Double-clicking this icon opens the check editor.

To know more about the Check Editor, see Using the Check Editor.

To know more about the Check Editor, see Using the Check Editor.

Performing a Global Analysis of Checks

This task explains how to perform an analysis of Knowledge Expert and Knowledge Advisor Checks. The scenario is divided into 2 major steps:

- parameters, formulas and checks are created,
- the checks analysis is run and the checks that failed are corrected.

To know more about the Global Analysis tool and the Check Report, see Using the Check Analysis Tool and Customizing Check Reports.

- For the check report to be correctly generated, go to **Tools->Options->General->Parameters and Measure ->Report Generation**, and select:

  - The Input XSL file under Input XSL. (An XSL file is provided by default. Click here to get a description of the generated XML file.)

  - The parameters you want to appear in the report under Report Content.

  - The Output directory under Output Directory.

1. Open the KwrCheckAnalysis.CATPart file. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench.
2. Create a parameter of Length type and assign it a formula. To do so, proceed as follows:
   - Click the icon. The formula editor opens.
   - Select Length in the scrolling list to define the type of the parameter, click the New parameter of type button, change the name of the parameter (Length in this scenario), and click the Add Formula button. The Formula Editor opens.

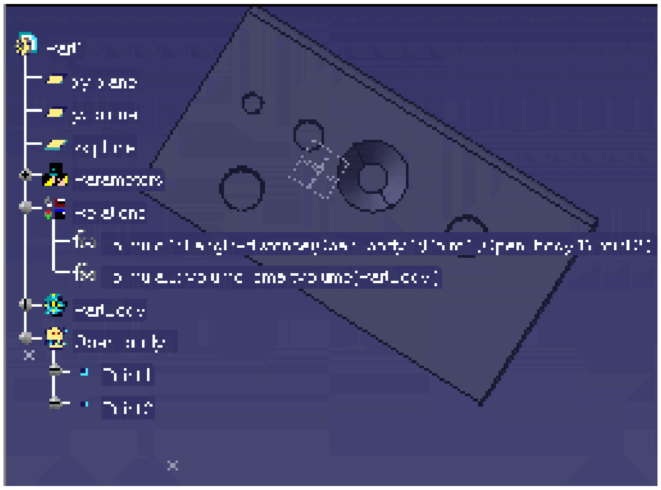
- Under Dictionary, select Measures, and double-click distance(Body,Body). Position the cursor before the coma and double-click Point.1 in the specification tree or in the geometrical area. Position the cursor after the coma and double-click Point.2 in the specification tree. Click **OK**.

3. Create a parameter of Volume type and assign it a formula. To do so, proceed as follows:

   - Click the  icon. The formula editor opens.

   - Select Volume in the scrolling list to define the type of the parameter, click the New parameter of type button, change the name of the parameter (Volume in this scenario), and click the **Add formula** button.

   - Under Dictionary, select Part Measures, and double-click smartVolume. Position the cursor between the parentheses and select PartBody in the specification tree. Click **OK**.



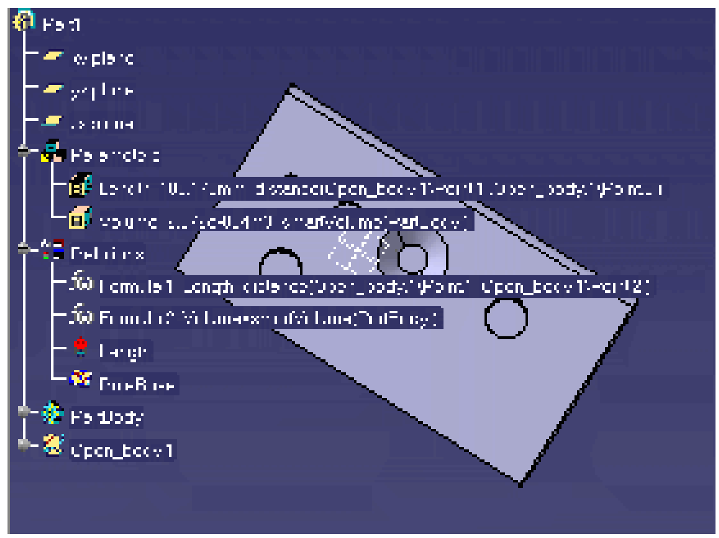
The parameters and the associated formulas are created (click the graphic opposite to enlarge it)

4. Access the Knowledge Advisor workbench, click the **Check** icon ( ), change the name of the check (Length in this scenario), and click **OK**. The Check Editor opens.

5. Enter the following script in the editor, then click **Apply** and **OK**.

```
Length > 150mm
```

The Knowledge Advisor Check is created (click the graphic opposite to enlarge it).



6. Access the Knowledge Expert workbench, click the Expert Check icon, and change the name of the check (HoleCheck in this scenario). The Expert Rule Editor opens.

7. In the Condition tab, enter the following script:

	H:Hole
Editor	H.Diameter > 15mm

8. Click the Correction tab, select VB Script in the scrolling list and enter the following script in

the editor:

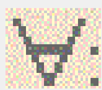
```
Dim aHole as Hole
Set aHole = H.parent.Item(H.Name)
Dim diam As Length
Set diam = aHole.Diameter
diam.Value = 16
MsgBox("Correction performed on "&H.Name)
```

9. In the Correction Comment field of the Correction tab, enter the following string, and click **OK**:

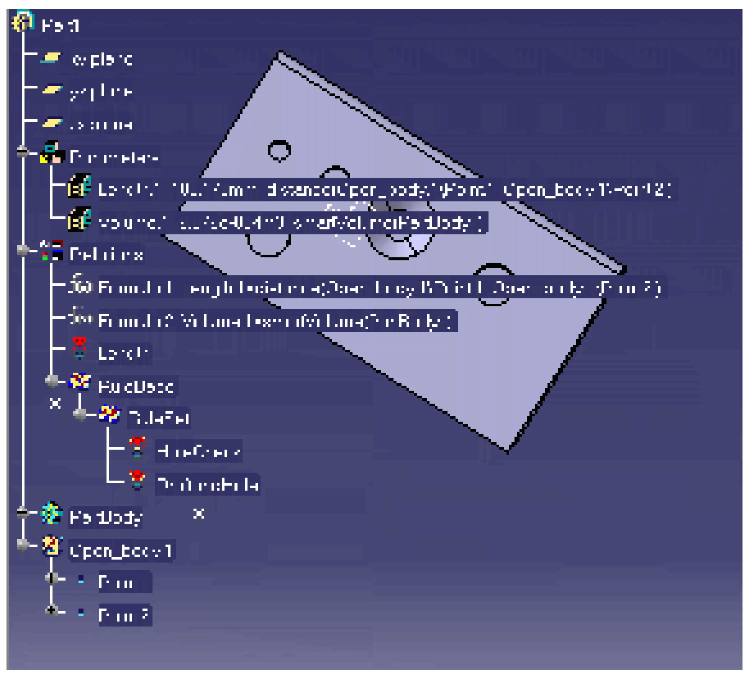
   Holes diameter should be greater than 15mm.

10. Select the Rule Base under the Relations node and click the Expert Check icon, change the name of the check (DraftandHole in this scenario), and click **OK**. The Expert Check Editor opens.

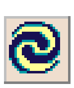
11. In the Condition tab, enter the following script, then click **Apply** and **OK**.

	H:Hole ; D:Draft
Editor	D.Activity AND H.Diameter > 12mm

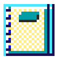
The checks are created (click the graphic opposite to enlarge it).



12. Click the  icon in the toolbar. The Global Analysis Tool opens.

13. Click the  icon to update the status of the checks. The Checks lights turn to red in the

specification tree.

14. Click the  icon. An xml page opens indicating the items that failed. To know more about this report, see Customizing Check Reports.

15. Click the  icon to launch the correction method specified when creating the Expert check (See step 9). The checks have been corrected.

Only the Advisor check (Length) could not be corrected: The value of the Length parameter is 100.175 mm (as indicated in the report) whereas it should be superior to 150mm (as indicated in the body of the check).

16. To correct the check, modify the value of the Length parameter. To do so, proceed as follows:

- Double-click Point.1 in the geometrical area. The Point definition window opens.
- In the H: field, change the value of the point to 150mm. Click **OK**. The light of the check turns to green indicating that the check is passed.

Using the Check Analysis Tool

The Global Analysis Tool is designed to manage Expert and Advisor checks wherever they may be located in the specification tree. It helps end-users understand the validation status of their designs and allows navigation by checks or violations and highlights failed components. The user can:

- Access information concerning failing items
- Gather information concerning objects and checks
- Perform automatic corrections if need be.

The Global Analysis tool can be accessed at the session level by clicking the icon in the toolbar. This icon provides the user with a simple Checks status:

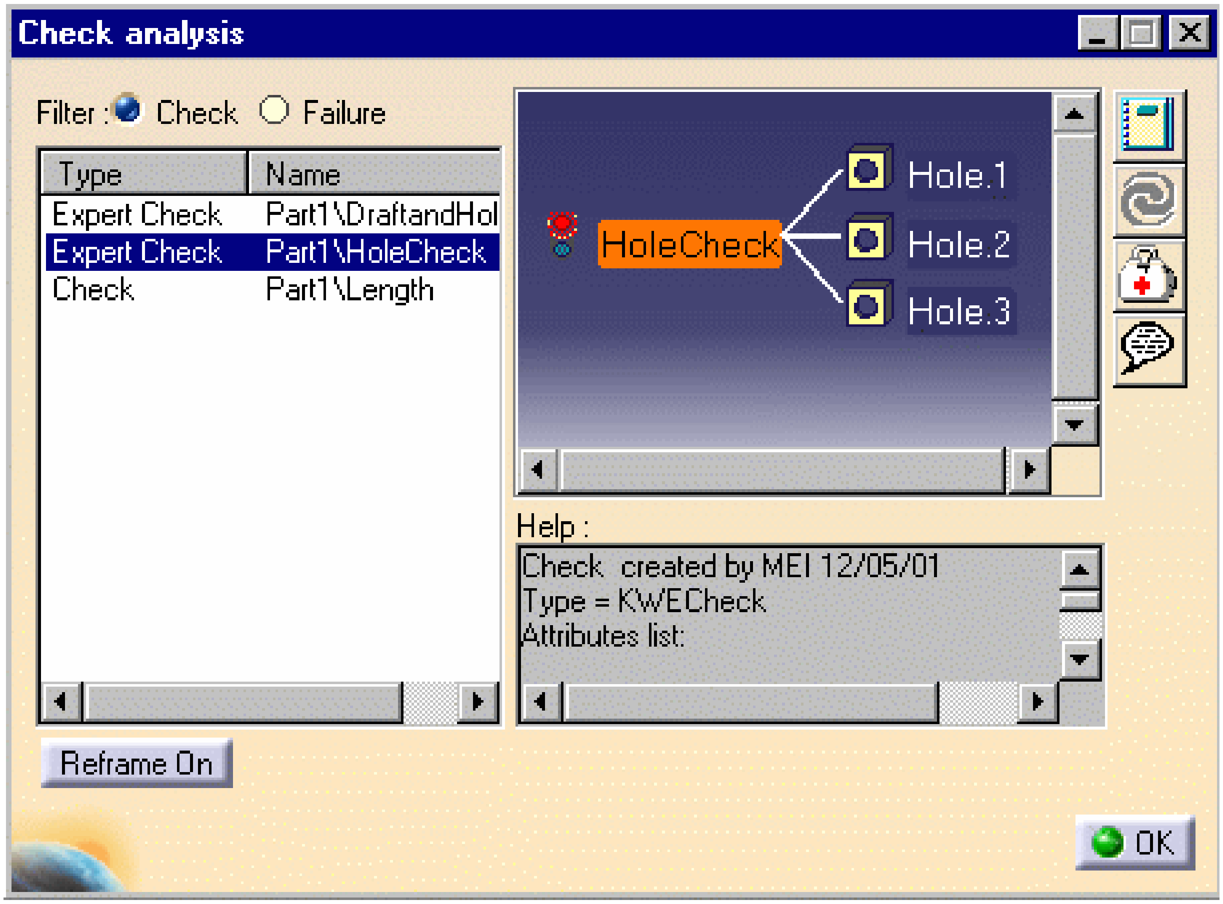
All the checks are updated and could be fired successfully.

The checks need to be updated.

All the checks are updated and at least one of them is incorrect.

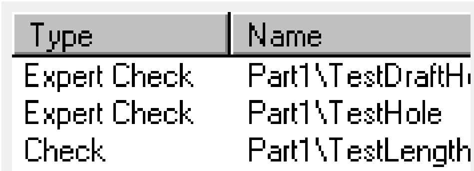
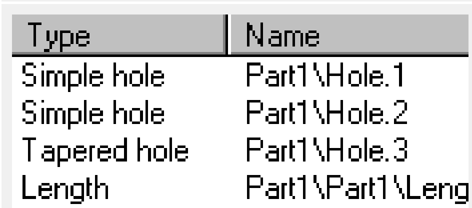
Check Analysis Tool Window

Click the icon in the toolbar to access the Check analysis window.



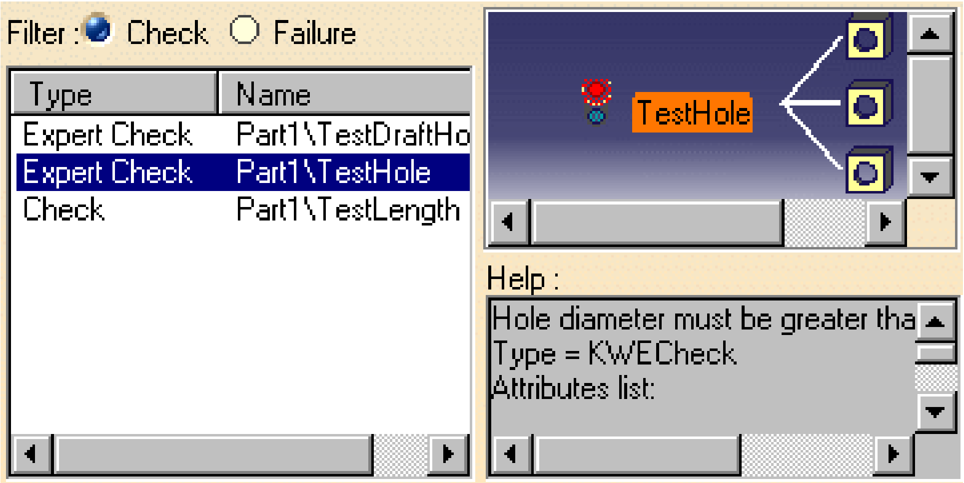
Filter section

This option enables the user to apply a filter to the checks or to the items that failed.

Check	Only the Expert and Advisor checks that failed when updating the check report are displayed.	
Failure	All the items that failed when updating the check report are displayed.	

Help section

To display the help section associated with each item of the list, double-click the desired item. The following view is displayed:



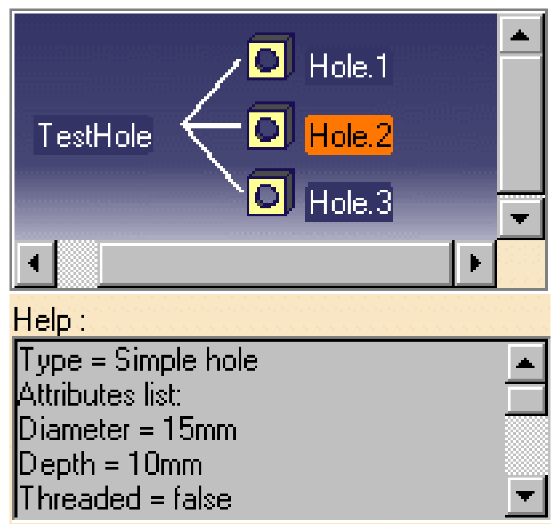
The check and the items that it controls are displayed in the view as well as its current status.

The items entered when creating the check are displayed:

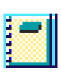
- Associated comments
- Type
- Attributes
- Variables
- Name
- Owner of the check...

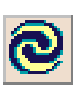
In the graphic above, the selected check is TestHole, it checks the holes of the CATPart file (3 of them do not pass the check because their diameters is not superior to 15mm), and the attributes are displayed corresponding to the data entered when creating the check.

Note that it is also possible to select the items associated to the check.
To do so, double-click the desired item in the view: The Help section shows the information concerning this item (see graphic opposite.)



Toolbar

 Click this icon to generate the customizable check report. To know more about the check report, see Customizing Check Reports.

 Click this icon to solve the checks created in your document.

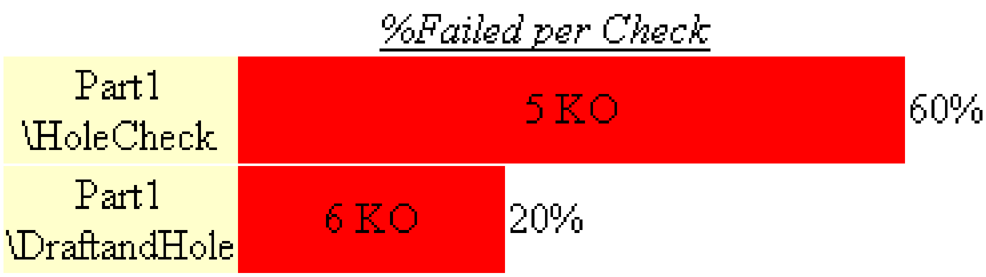
 Click this icon to launch the correction method specified in the Check Editor when creating the check. For an example, see Performing a Global Analysis of Checks.

 Click here to display the URL associated to the object, or to assign an URL to an object. To know more, see Associating URLs and Comments with Parameters or Relations.

Introducing the Default Check Report

The default check report presents the Expert and Advisor checks that failed.

1. Part1\Length
2. Part1\HoleCheck
3. Part1\DraftandHole



This panel lists the checks that failed and presents a percentage of the failed items per Expert Check.

Advisor Checks report

Check advisor: Part1\Length

Body

```
/*Check created by MEI 11/29/01*/
Length > 150mm
```

This check operates on :

- *Part1\Length = 100.175mm*

The Advisor checks panel lists the Advisor checks that failed and shows the following elements:
- the body of the check (Length>150mm here)
- the item(s) on which the check operates (here, the Length formula).

Expert Checks report

Part1\HoleCheck

Comment					
Input	Part1 \Hole.1	Part1 \Hole.2	Part1 \Hole.3	Part1 \Hole.4	Part1 \Hole.5

	Part1\Hole.1
	Part1\Hole.2
	Part1\Hole.3
	Part1\Hole.4
	Part1\Hole.5

The Advisor checks panel lists the Expert checks that failed and shows the following elements:
- the Input items checked by the check operation.
- the item(s) that failed (here Hole.1, Hole.2, and Hole.3).

Remember that this report should not be used to generate macros or other files. It is provided as information only.

Customizing Check Reports

The reports generated by the Global Check Analysis Editor can be customized.

You can choose to display a xml or a html report.

Displaying a HTML report

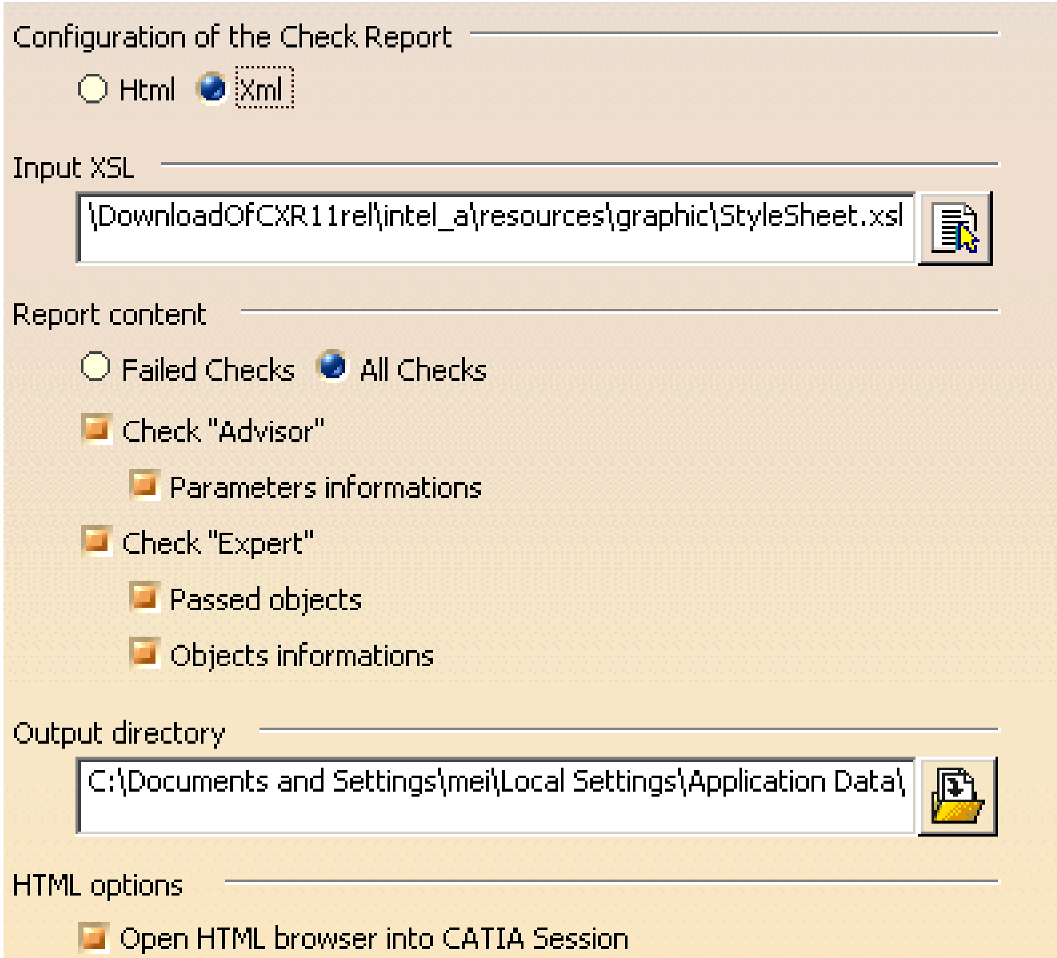
To generate a html report when performing the check analysis, go to **Tools->Options->General->Parameters and Measure** and select the **Report generation** tab. Select **Html** in the Select **Configuration of the check report** area.

In this case, only the **Check Advisor**, the **Check expert** and the **Passed objects** options are available in the Report content area. You can specify the output directory containing the generated HTML report in the **Select output directory** field.

Select **Html** if you use a Netscape browser.

Displaying a XML report

To display a XML report when performing the check analysis, go to **Tools->Options->General->Parameters and Measure** and select the **Report generation** tab. Select **Xml** in the Select **Configuration of the Check Report** area. The following window opens:



The **Report generation** tab is made up of 4 different fields: The Input XSL, the Report Content, the Select output directory, and the HTML options fields.

Input XSL field

This field enables the user to select the XSL style sheet that will be applied to the generated XML report. The StyleSheet.xsl file is the default XSL file, but you can use your own template.

Report content field

Failed Checks		If checked, the generated report will contain information about the failed checks only.
All Checks		If checked, the generated report will contain information about all the checks contained in the document.
Check advisor		If checked, the generated report will contain information about all the Knowledge Advisor checks contained in the document.
	Parameters information	If checked, the generated report will contain information about the parameters of the Advisor checks.

Check expert		If checked, the generated report will contain information about all the Knowledge Expert checks contained in the document.
	Passed objects	If checked, the generated report will contain information about the objects that passed the Expert checks as well as information about the parameters of these objects (diameter, depth, pitch,...).
	Objects information	If checked, the generated report will contain information about all the objects contained in the Expert checks as well as information about the parameters of these objects (diameter, depth, pitch,...).

Output directory field

This field enables the user to select the output directory containing the generated XML report.

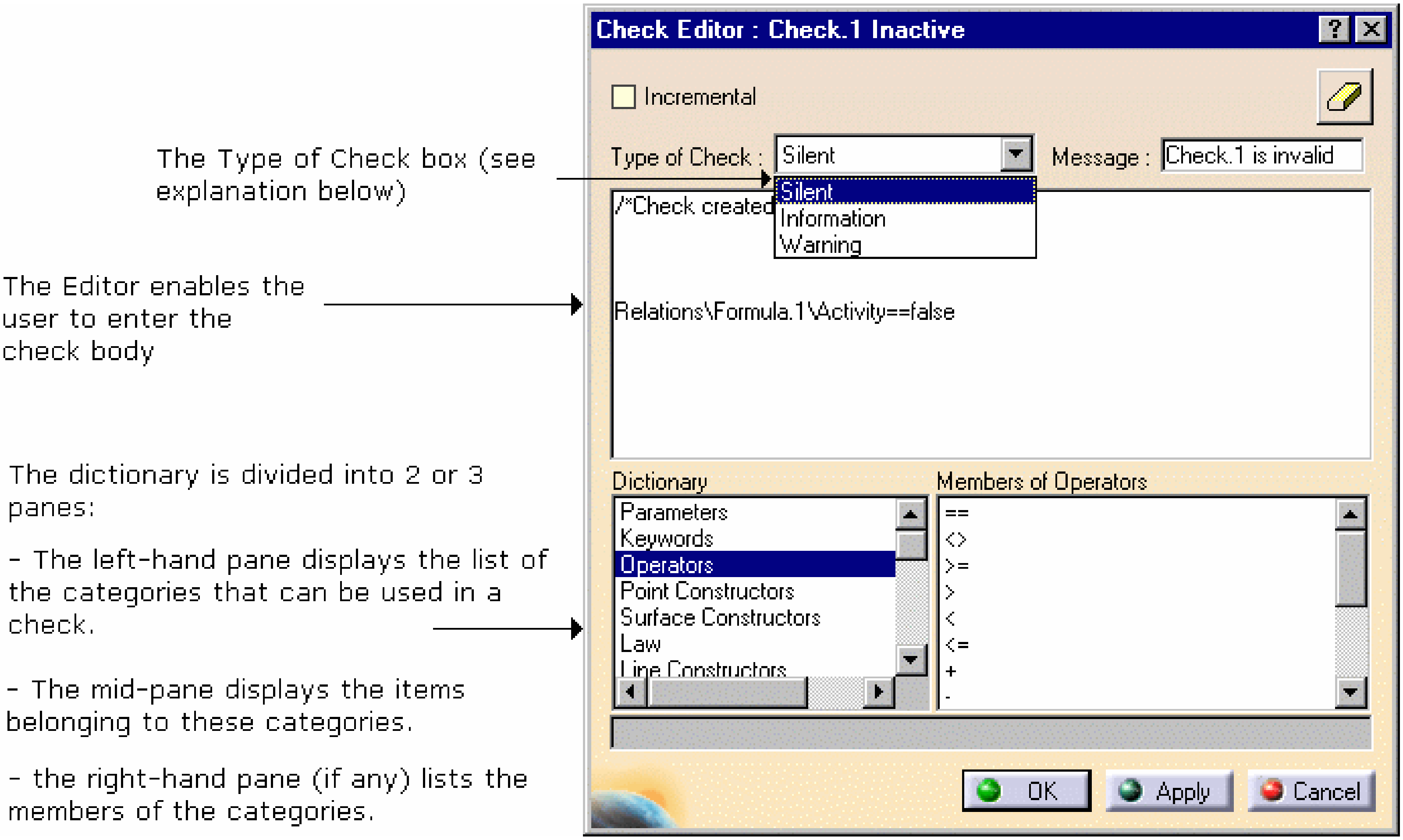
HTML options field

This option is available for Windows only. It enables the user to define if the report will be opened in a *CATIA* session (in this case, the check box should be checked) or if it will be opened in an Internet Explorer session (in this case, the check box should remain unchecked.)

Note that it is highly recommended not to use this report as a basis for macros or for other applications. It is only provided for information purposes.

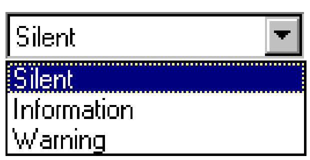
Using the Check Editor

The Check Editor is intended to help the user enter the check body using the Dictionary. It is made up of 3 different areas:



Three different types of checks can be used:

- The silent checks
- The information checks
- The warning checks



Depending on the type of check and the result of the check, you will be warned as follows:

	Check verified	Check not verified
Relation icon in the specification tree		
Silent check	no message displayed	no message displayed
Information check	no message displayed	 the message specified at check creation is displayed in an information box

Warning check	no message displayed		the message specified at check creation is displayed in a warning box.

In the Check Editor, you can:

- Restrict the list of parameters displayed in the dictionary: To do so, go to the specification tree, simple click the feature you want to display the parameters for. If the 'Incremental' option is selected, only the first level of parameters located right below the selected feature are displayed. If not, all the parameters at all levels are displayed.
- Insert the feature definition in a check: To do so, go to the specification tree, and double-click the feature you want to insert the definition for.
- Check whether the check syntax is correct by clicking **Apply**.
- Erase the contents of the edition window by clicking the icon.
- Add the check to the document by clicking **OK**.

To know more about the Dictionary, see Using the Dictionary. To know more about checks, see Creating a Check.

Working with the Reaction Feature

Click the Reactions icon to create a script specifying how to change some feature attributes when an event occurs.

The reaction is a feature that reacts to events on its source(s) by triggering an action. It is designed to cope with the rules and the behaviors limitations and to create more associative and reactive design.

A reaction is a feature that reacts to events

The source can be:

- A selected feature (or a list of features)
- A parameter (result of a test)

Events can be:

- General events on objects (creation, deletion, update, drag and drop, attribute changes) and parameter value changes.
- Specific events such as a power copy or a UDF instantiation/update.
- Insert/Replace component
- Object Drag and Drop

A reaction is similar to a rule in the fact that:

- It is stored in the model.
- It reacts to changes and can trigger modifications.
- It also references other objects and parameters in the document and supports the replace mechanism.

But

- Reaction features provide a better control over when the action has to be fired.
- Reactions enable the user to perform more complex actions. Since you have better control when the action is triggered, and as you're not constrained by the update mechanism limitations, you can use all the power of any Visual Basic API (in *CATIA* but also in other automation applications...), and a Visual Basic macro can be called with arguments from an action.
- Reactions can be written to customize the update mechanism (to optimize user features, for example).

- Reactions can react to user actions (instantiation of a user defined feature), insertion of a component in an assembly, modification of a parameter...
- Reactions can be stored in the model and can be integrated in the definition of a power copy or user feature.

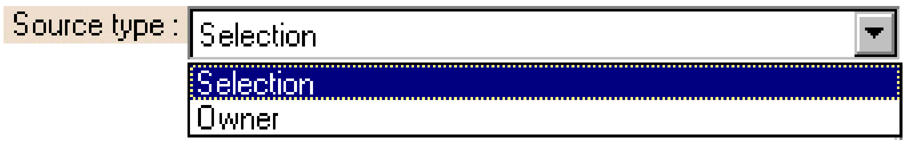
The reaction feature is not integrated to the update contrary to the rule. If a reaction is launched when updating the document, it may impact the document.

Using the Reaction Feature Window

You can access the Reaction window by clicking the  icon in the Knowledge Advisor workbench.

The Reaction window is made up of 3 major fields: The Source Type field, the Source field and the Action field.

Source Type



- **Selection** enables the user to manually select one or more items in the specification tree or in the geometrical area. These items will be displayed in the **Sources** field.
- **Owner** enables the user to link the action with a feature of the geometry or of the specification tree (see Using the Knowledge Advisor Reaction Feature: DragAndDrop Event where the reaction feature is linked with a Hole, for example).
  To link the reaction with an object of the geometry, click the **Destination** field and select an object in the specification tree or in the geometry.

Sources Field

A reaction is a feature that reacts to events (see Available events below) on an object called the source and that triggers an action.

The Sources field displays the selected items with which the reaction will be linked (only available if the **Selection** Source type is selected.)

Available events

The events available in this scrolling list depend on the source type selected in the **Source type** field. The reaction will be fired when one of the events detailed below happens.

Available Events	Explanation
AttributeModification	The reaction is fired because of a change in an attribute state. Only available if the **Selection** option is selected.
BeforeUpdate	The reaction is fired before a feature is updated.
DragAndDrop	The reaction is fired after a feature is dragged and dropped.
Insert	The reaction is fired when a feature is inserted.

Inserted	The reaction is fired after a feature is inserted.
Instantiation	The reaction is fired when a user feature is instantiated.
Remove	The reaction is fired when a feature is removed.
Update	The reaction is fired right after a feature is updated.
ValueChange	The reaction is fired because of a parameter value change. Only available if the **Selection** option is selected.
FileContentModification	The reaction is fired each time the file associated to the design table is modified.

Action Field

The action is triggered by a reaction that reacts to events on an object. This field enables the user to select the language in which he wants to write the action (VB or the Knowledge Advisor language) and to edit the action.

Edit action button

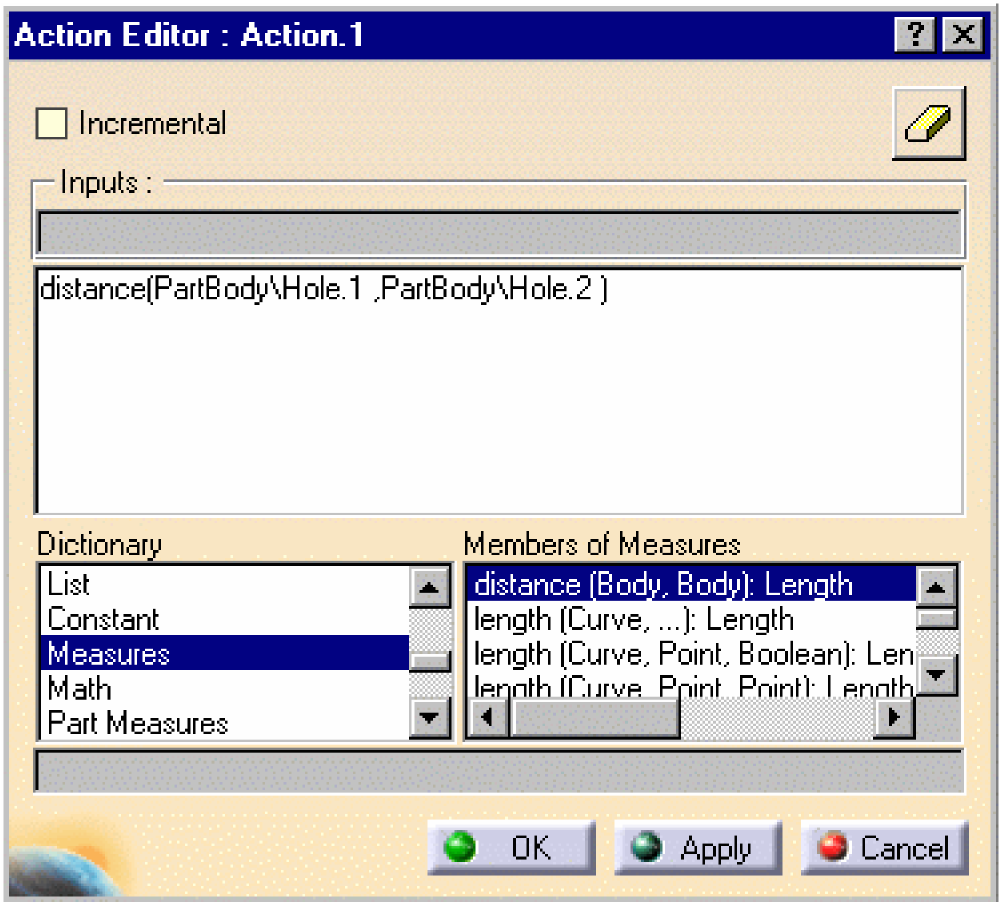
Click this button to access the Action Editor.

Action Editor

The action editor displayed depends on the language selected in the **Action** field.

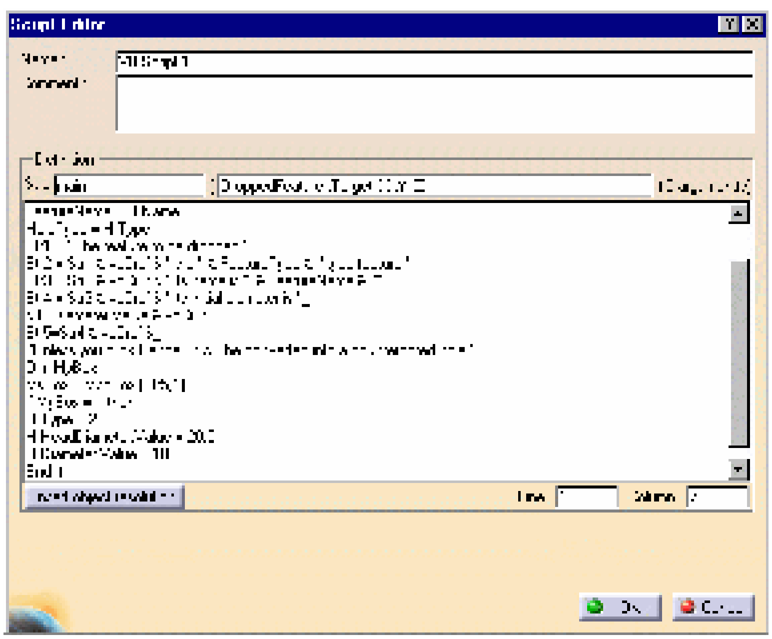
If **Knowledgeware action** is selected, the window below is displayed.

The Edition pane enables the user to enter the body of the action.



The **Dictionary** is divided into 2 or 3 panes depending on the selected category:
- The left-hand one displays the categories that can be used in an action.
- The middle one lists the objects belonging to the selected category.
- The right-hand one displays the members of the selected category.

If **VB action** is selected, the window below is displayed.



- The **Name** field enables the user to enter a name for the VB script.

- The **Comment** field enables the user to enter a comment associated to the VB script.

- The **Editor** enables the user to enter the VB script.

- The **Insert object resolution** button enables the user to select an object in the specification tree or in the geometry and to automatically add its resolution to the script.

(Click the graphic opposite to enlarge it.)

Creating a Reaction: DragAndDrop Event
Creating a Reaction: Insert Event
Creating a Reaction: Inserted Event

Creating a Reaction: DragAndDrop Event

This task explains how to use the DragAndDrop event in a reaction feature. In the scenario below, the user drags and drops a hole, which fires a reaction.

a) The following information are displayed in a VB box when the rule is fired:

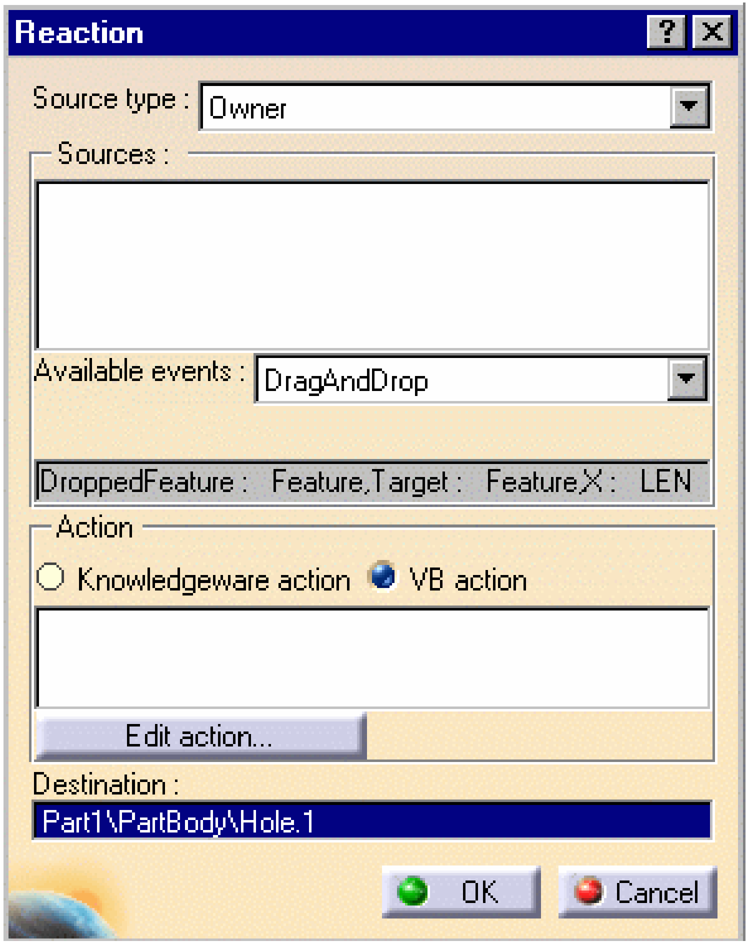
- The type of feature.
- Its name as well as its initial diameter.

b) The user is prompted to click **OK** to convert the hole into a counterbored one or to click **Cancel** to skip the conversion.

Note that this task could be carried out in the past by using the Behavior feature which has been replaced with the Reaction feature.

For more information about Reaction features, see Working with the Reaction Feature.

1. Create .CATPart file and a pad with a hole or open the KwrReactionPad.CATPart file.
2. Access the Knowledge Advisor workbench and click the Reaction icon ( ) to create a reaction. The reaction dialog box opens.
   - In the **Source type** field, select **Owner** for the Reaction to be applied to the hole selected in the **Destination** area (see below).

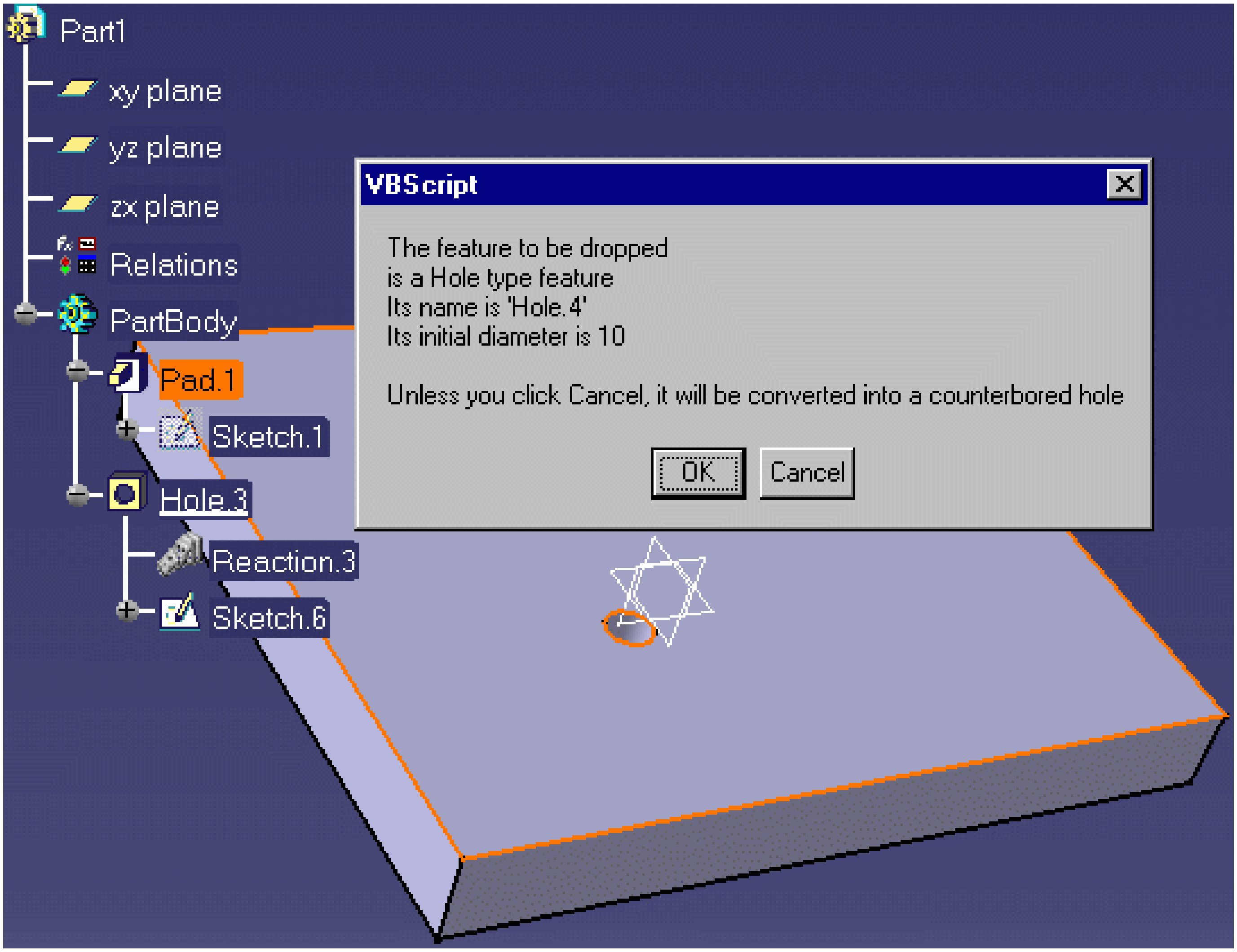


- In the **Available events** list, select **DragAndDrop** for the reaction to occur when the hole is dragged and dropped.
- In the **Action** field, select **VB action**, for the user to write the action in VB.
- Click the **Destination** area in the Reaction dialog box and select Hole.1 in the specification tree.

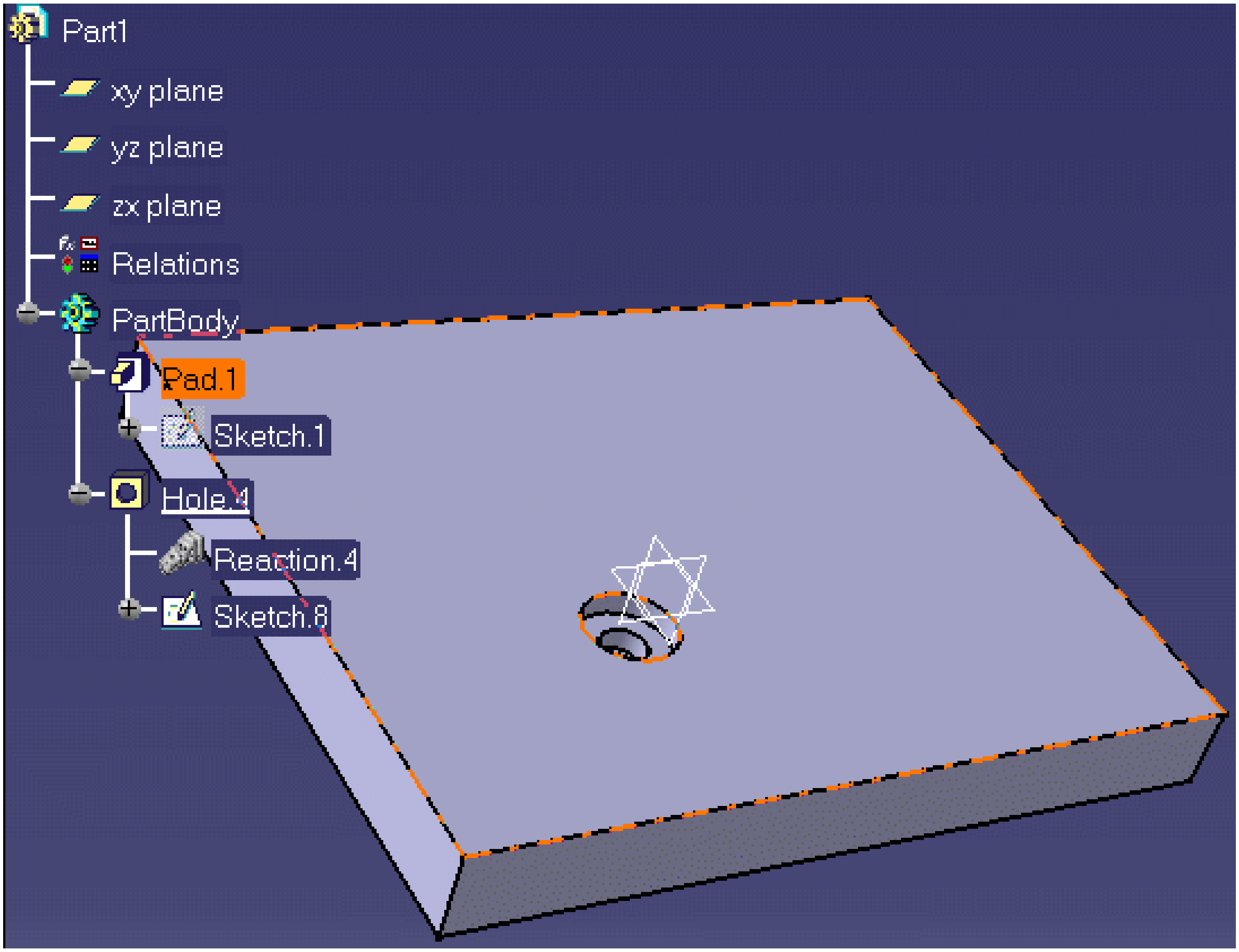
3. Click the **Edit Action...** button, paste the following script in the editor, and click **OK twice**:

```
Set H = DroppedFeature.Parent.Item(DroppedFeature.Name)
Dim FeatureType, FeatureName, HoleType
FeatureType = TypeName(H)
FeatureName = H.Name
HoleType = H.Type
Str1 = "The feature to be dropped"
Str2 = Str1 & vbCrLf & "is a " & FeatureType & " type feature"
Str3 = Str2 & vbCrLf & "Its name is '" & FeatureName & "'"
Str4 = Str3 & vbCrLf & "Its initial diameter is "_
& H.Diameter.Value & vbCrLf
Str5=Str4 & vbCrLf &_
"Unless you click Cancel, it will be converted into a counterbored hole"
Dim MyBox
MyBox = MsgBox (Str5,1)
if MyBox = 1 then
H.Type = 2
H.HeadDiameter.Value = 20.0
H.Diameter.Value = 10.0
End If
```

4. Access the Part Design workbench. In the geometry, select the hole and drag and drop it. The following dialog box appears:



5. Click **OK**. The hole is converted into a counterbored hole (see graphic below).



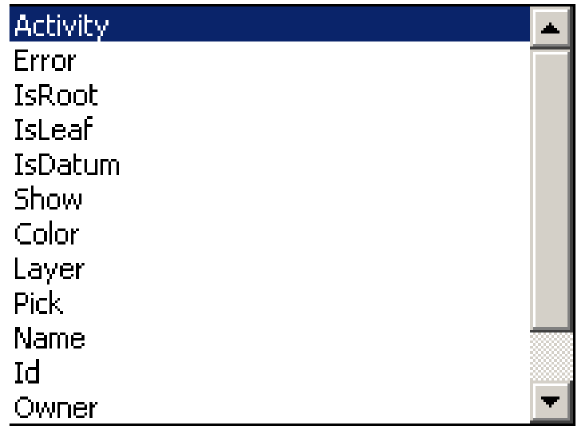
To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Reaction: AttributeModification Event

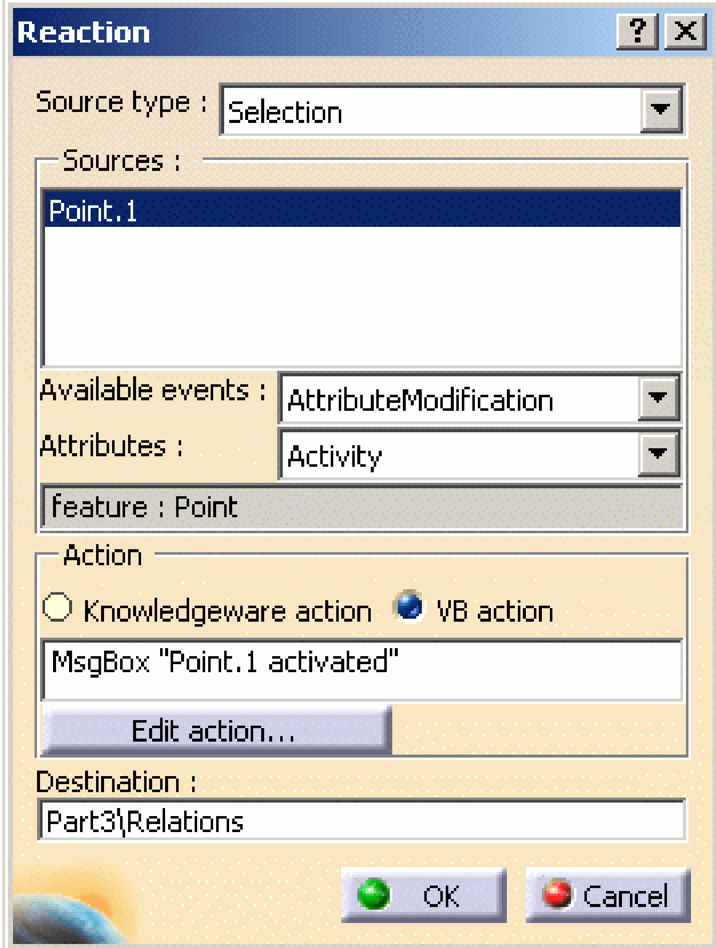
This task explains how to use the AttributeModification event in a reaction feature. In the scenario below, the user creates a reaction based on the activity (activated or deactivated) of a point. If the point is activated, a message box displays.

This reaction is designed to react to attributes modification (color, state, name, thickness...).

Note that attribute modification events are not available for all features. If it is the case with the feature you have selected, select the attribute in the **Select a source...** window and use the ValueChange event. The reaction will be launched when the attribute is modified.

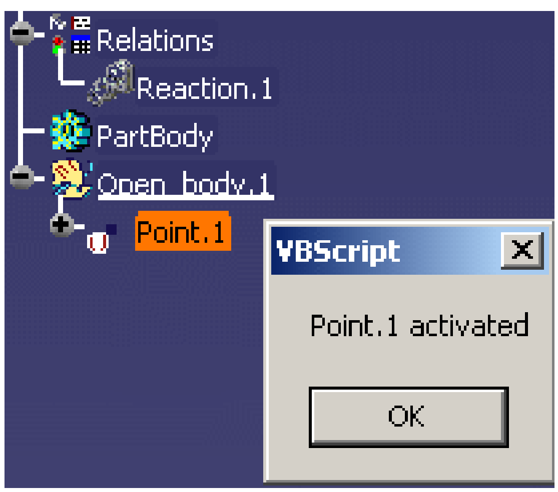


1. Open the KwrAttributeModificationEvent.CATPart file. This file contains a point that is deactivated.
2. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench and click the **Reaction** icon ( ) to create a reaction. The reaction dialog box opens.
   - In the **Source type** field, select **Selection** and select Point.1 in the specification tree.
   - In the **Available events** scrolling list, select **AttributeModification**.
   - In the **AttributesLabel** scrolling list, select **Activity** for the reaction to be launched when the point is activated or deactivated.
   - In the **Action** field, select **VB action** and enter the following text into the editor: **MsgBox "Point.1 activated"**



- Click **OK** when done. The reaction is created and displays below the Relations node in the specification tree.

3. Double-click Point.1 in the geometry to access the Generative Shape Design workbench.
4. Right-click Point.1 and select the **Point.1 object->Activate** command. The following dialog box appears:



5. Click **OK.** Point.1 is activated.

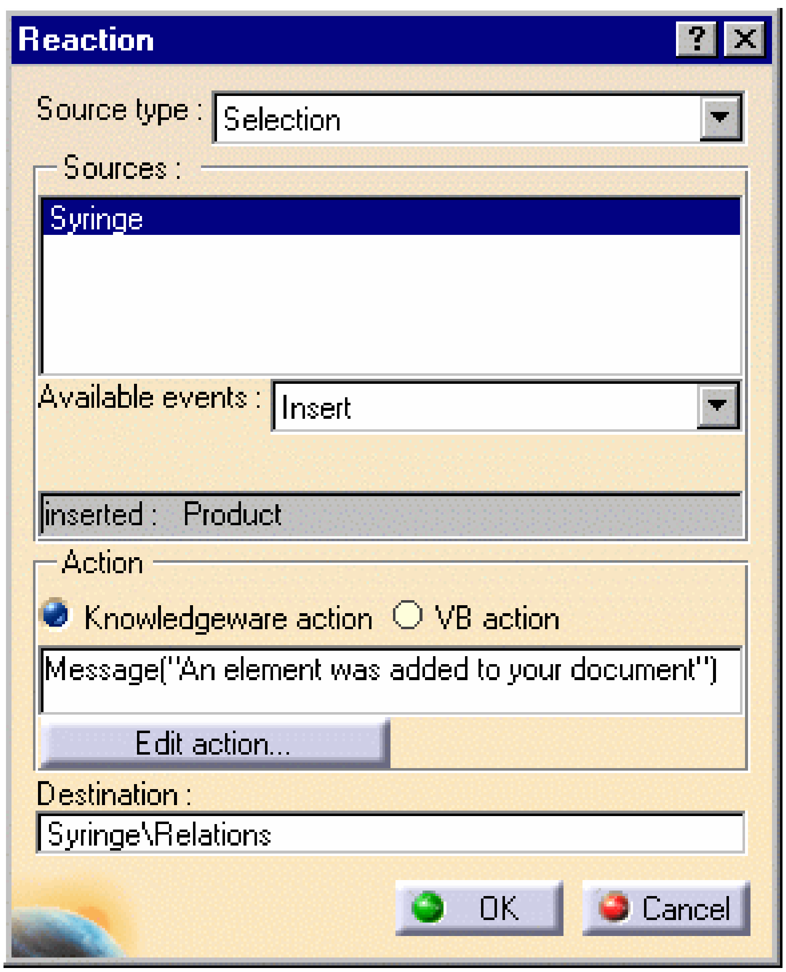
To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Reaction: Insert Event

This task explains how to use the Insert event in a reaction feature. In the scenario below, the user inserts an element into the CATProduct document, which displays a message.

1. Open the KwrSyringeAssembly.CATProduct file.

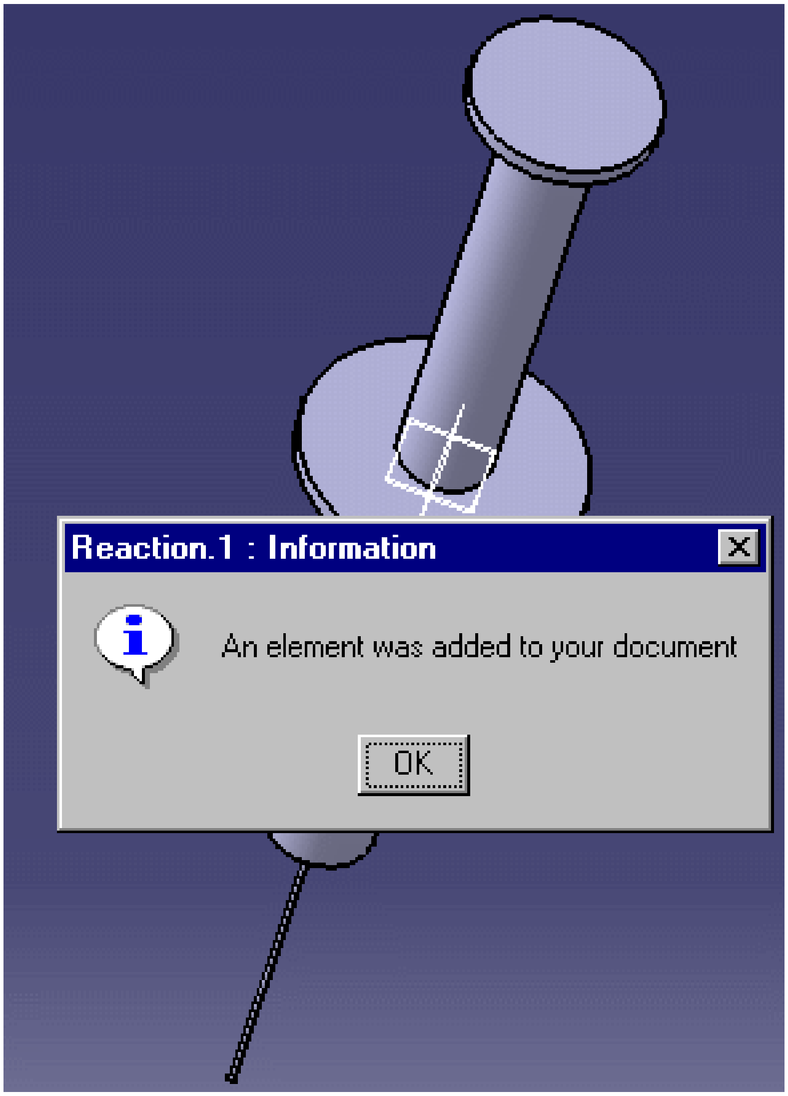
2. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench and click the Reaction icon ( ) to create a reaction. The reaction dialog box opens.



- In the **Source type** field, select **Selection** for the Reaction to be applied to the element you select and select the syringe in the specification tree).
- In the **Available events** list, select **Insert** for the reaction to occur when an item is inserted into the CATProduct.

- In the **Action** field, select **Knowledgeware action** and enter the following message: Message("An element was added to your document"). This message will be displayed each time you insert a new component into the CATProduct.
- Click **OK**. A reaction is added to the Relations node in the specification tree.

3. Double-click the root of the specification tree, select the **Insert->Existing Component...** command and click the root of the specification tree. The File selection dialog box opens.
4. Select the KwrSyringePiston.CATPart file and click **Open**.
5. The new element is inserted and the reaction is fired. The following message displays:

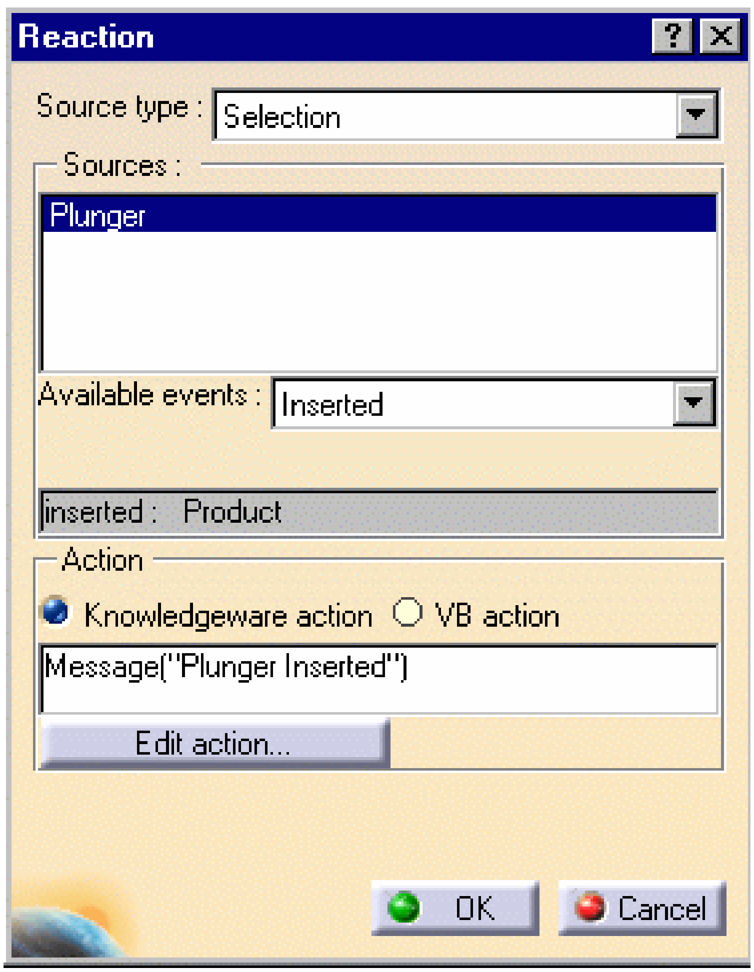


To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Reaction: Inserted Event

This task explains how to use the Inserted event in a reaction feature. In the scenario below, the user inserts an element into the CATProduct document, which displays a message.

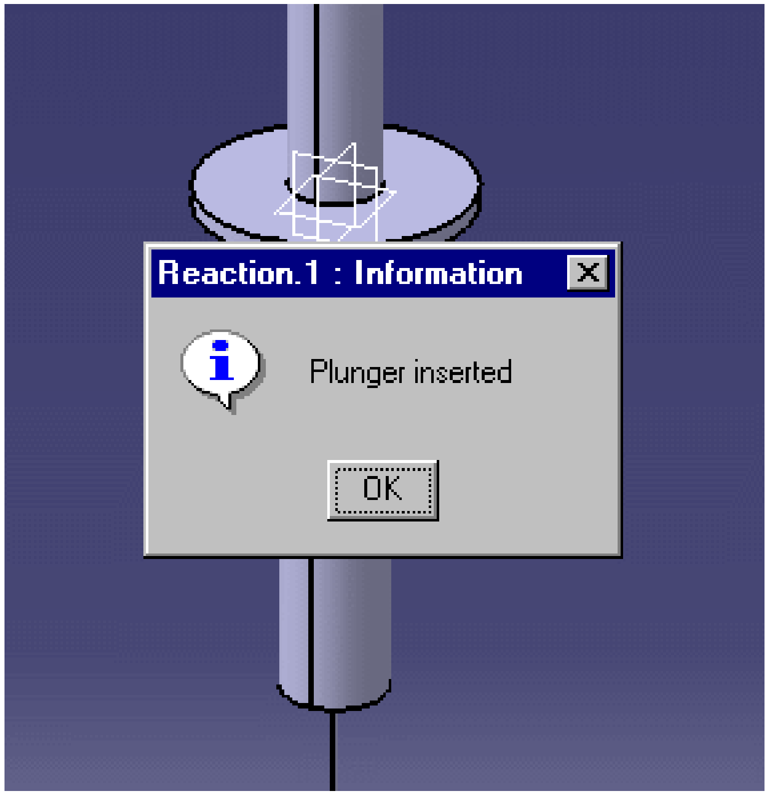
1. Create a CATProduct file called Container.CATProduct and insert the KwrSyringeContainer.CATPart file by using the **Insert->Existing Component...** command. Save your file and close it.
2. Create a CATProduct file called Plunger.CATProduct, rename the root of the specification tree to Plunger, and insert the KwrSyringePiston.CATPart file by using the **Insert->Existing Component...** command. Close the file.
3. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench and click the Reaction icon ( ) to create a reaction. The reaction dialog box opens.



- In the **Source type** field, select **Selection** for the Reaction to be applied to the element you select (plunger in this example).
- In the **Available events** list, select **Inserted** if you want the action to be launched when the Plunger is inserted into the CATProduct.

- In the **Action** field, select **Knowledgeware action** and enter the following message: Message("Plunger inserted"). This message will be displayed after the plunger is inserted.
- Click **OK**. A reaction is added to the Relations node in the specification tree.
- Save the file and close it.

4. Save the file and close it.
5. Open the Container.CATProduct file, select the **Insert->Existing Component...** command. The File Selection dialog box opens. Select the Plunger.CATProduct file and click **Open**. The message specified step 3 displays.



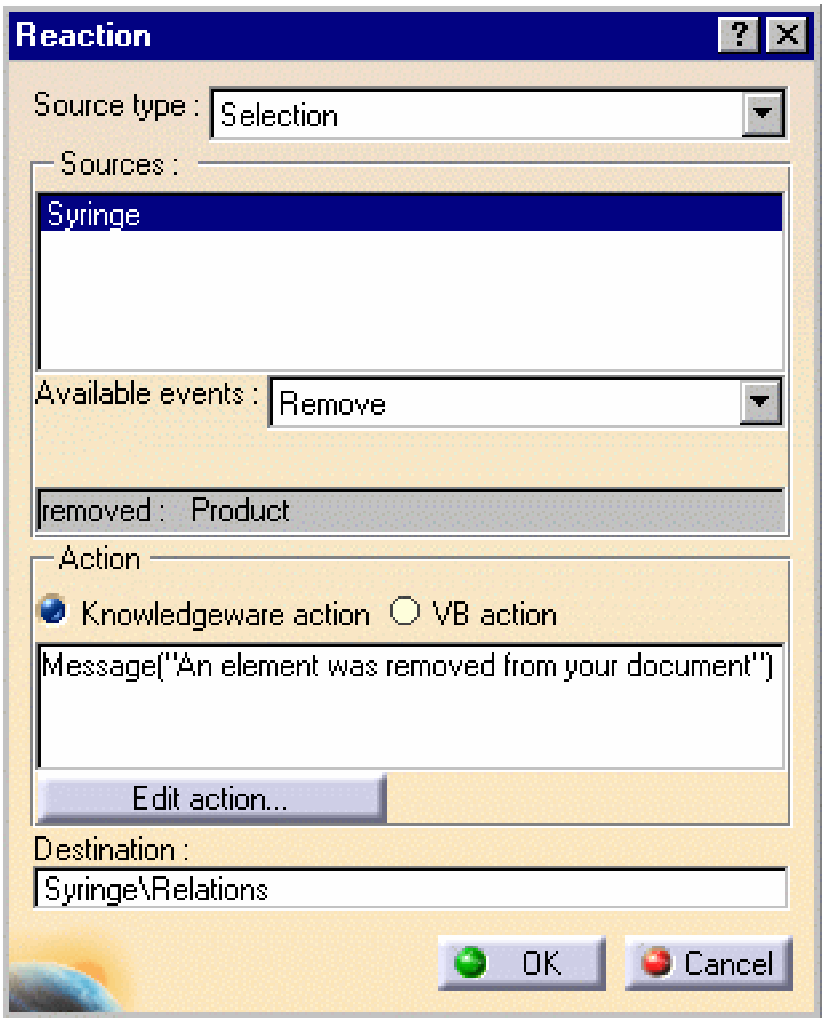
To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Reaction: Remove Event

This task explains how to use the Remove event in a reaction feature. In the scenario below, the user removes an element from the CATProduct document, which displays a message.

1. Open the KwrSyringeAssembly2.CATProduct file.

2. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench and click the Reaction icon ( ) to create a reaction. The reaction dialog box opens.

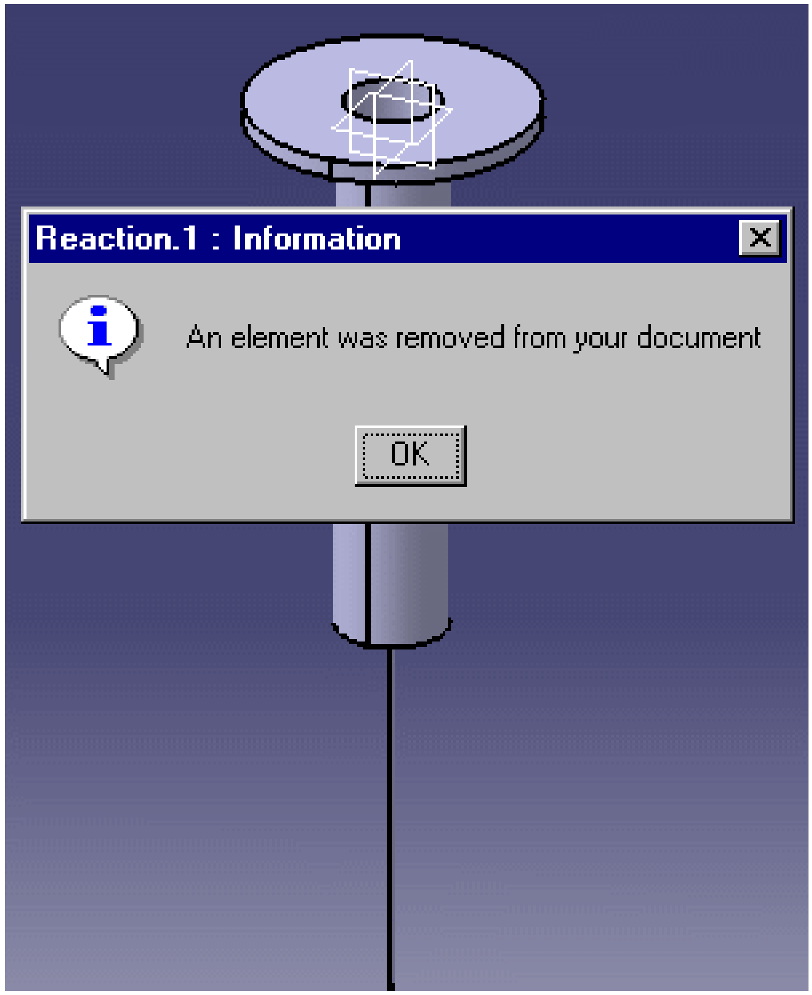


- In the **Source type** field, select **Selection** for the Reaction to be applied to the element you select (Syringe in this example).
- In the **Available events** list, select **Remove** for the reaction to occur after an item is removed from the CATProduct.

- In the **Action** field, select **Knowledgeware action** and enter the following message: Message("An element was removed from your document"): This message will display when you remove a component from the CATProduct.

  - Click **OK**. A reaction is added to the Relations node in the specification tree.

3. Double-click the root of the specification tree, right-click the Syringe piston in the specification tree, and select **Delete**. The following message displays.



To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Reaction: BeforeUpdate Event

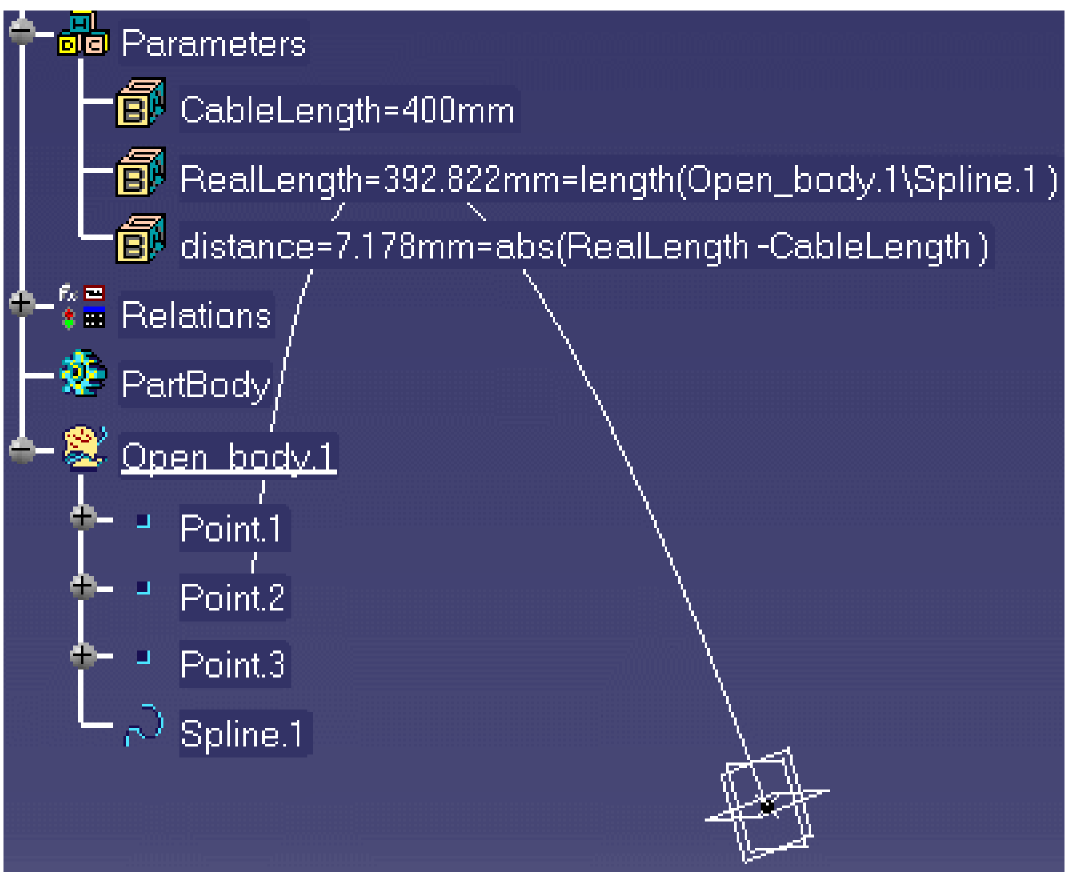
This task explains how to use the BeforeUpdate event in a reaction feature. In the scenario below, the user optimizes the position of a point each time he modifies the length of the cable (spline). The user creates his geometry and inserts all the components in a User Defined Feature (UDF).
This UDF contains the geometry of a cable going through 3 points:

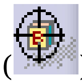
- The two points located at both extremities are to be specified in input.
- The coordinate of the third point is optimized in order to reach a target length for the cable.

This target parameter is a published parameter of the UDF. The optimization is launched by using a VBMacro with argument, called in a Reaction to the "Before Update" event of the UDF.

This scenario requires the Product Engineering Optimizer product.

1. Open the KwrEvent_BeforeUpdate.CATPart file: It contains 3 points and a spline (called cable in this scenario).



2. From the **Start->Knowledgeware** menu, access the **Product Engineering Optimizer** workbench and click the **Optimize** icon (). The Optimization window opens.

3. Enter the following data in the Optimization window:

Problem tab		
	Optimization type	**Minimization**
	Optimized parameter	**distance**
	Free parameters	**Geometrical Set.1\Point.3\Z**
	Algorithm	**Simulated Annealing-Convergence speed**
	Termination criteria	Maximum number of updates: **100** Consecutive updates without improvements: **20** Maximum time (minutes): **5**
Constraints tab		
	New constraint	**Geometrical Set.1\Point.3\Z` - max (Geometrical Set.1\Point.2\Z) ,Geometrical Set.1\Point.1\Z) <= 0mm**

4. Click **OK** in the opening dialog box, click **Run optimization.**

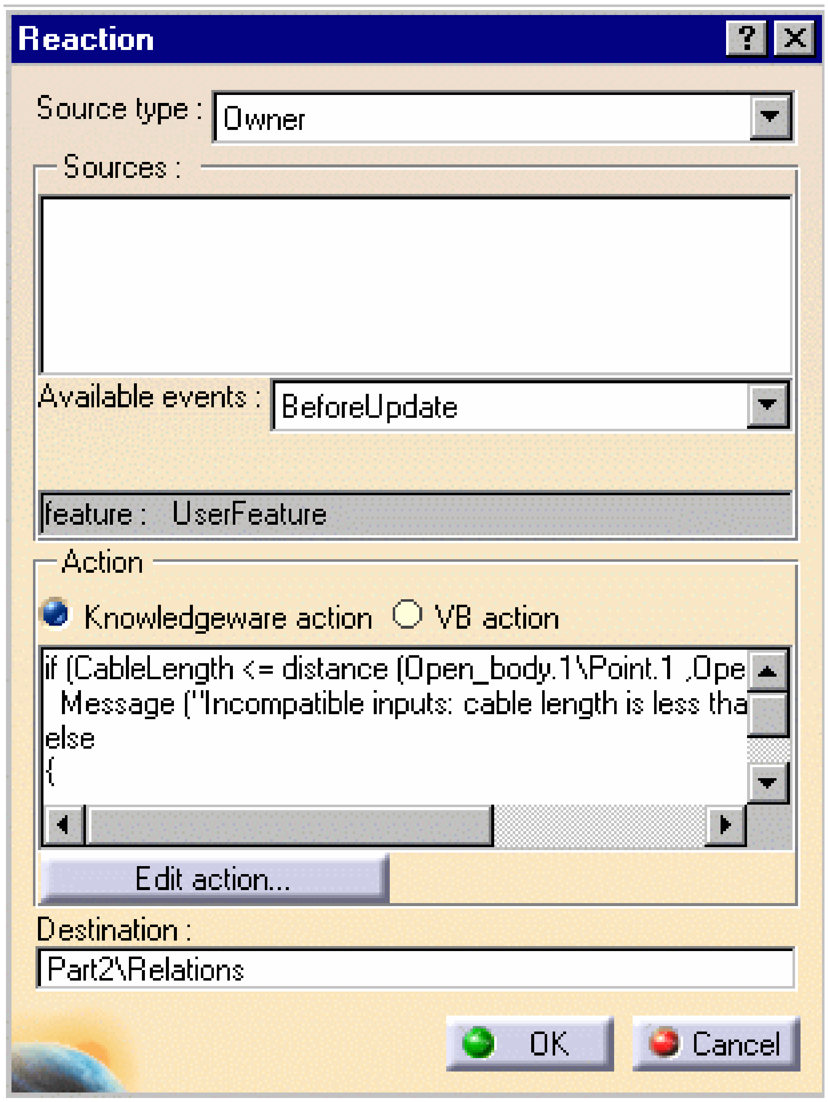
5. Select an output file and click **Save**.

6. Click **OK** once the optimization process is over.

7. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench and click the Macros with argument icon (). The Script Editor opens. Enter the following data in the editor and click **OK**:

Argument	**optim**
Script body	**optim.Run false**

8. Click the Reaction icon (). The Reaction dialog box opens.



- In the **Source type** field, select **Owner**.
- In the **Available events** list, select **BeforeUpdate**.
- In the **Action** field, select **Knowledgeware action**.

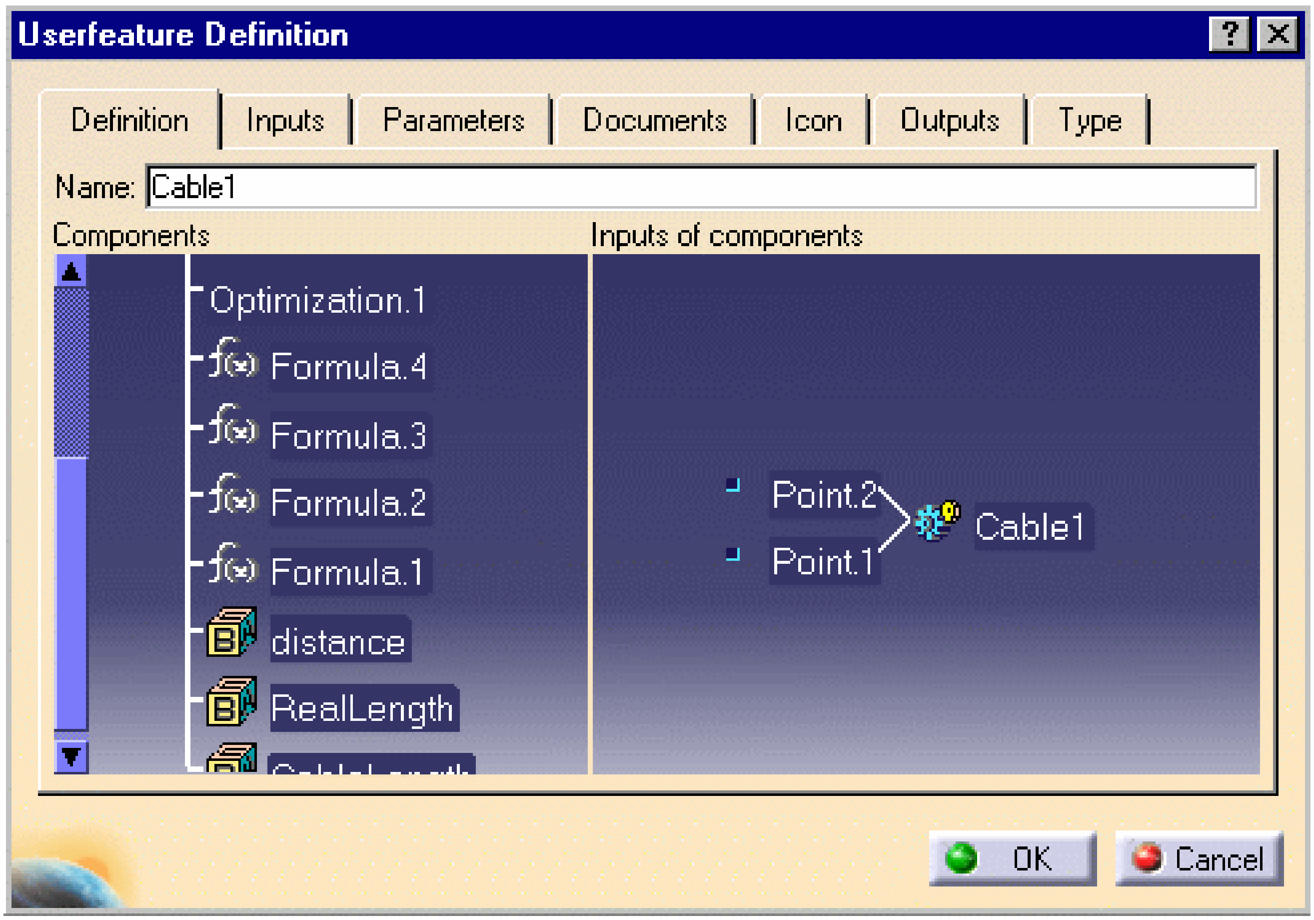
9. Click the **Edit action...** button, paste the following script in the editor, and click **OK** twice:

```
if (CableLength <= distance (Geometrical Set.1\Point.1 ,Geometrical Set.1\Point.2 ))
Message ("Incompatible inputs: cable length is less than distance between points!")
else
{
`VB Script.1` .Run(Relations\Optimizations.1\Optimization.1 )
}
```

10. Double-click the root of the specification tree and select the **Insert->UserFeature->UserFeature Creation...** command. The UserFeature Definition window opens.

11. In the **Name** field, enter the name of the UDF: **Cable1** in this scenario.

12. Select the Spline, Point3, Reaction.1, VB Script.1, Optimization.1, the 4 formulas, and the parameters: they are displayed in the UserFeature definition window (see below.)



Note that the UDF becomes the owner of the reaction. This reaction will be fired before the update of the UDF instance.

13. Click the **Parameters** tab, select CableLength, click the **Published name** check box

and click **OK**.

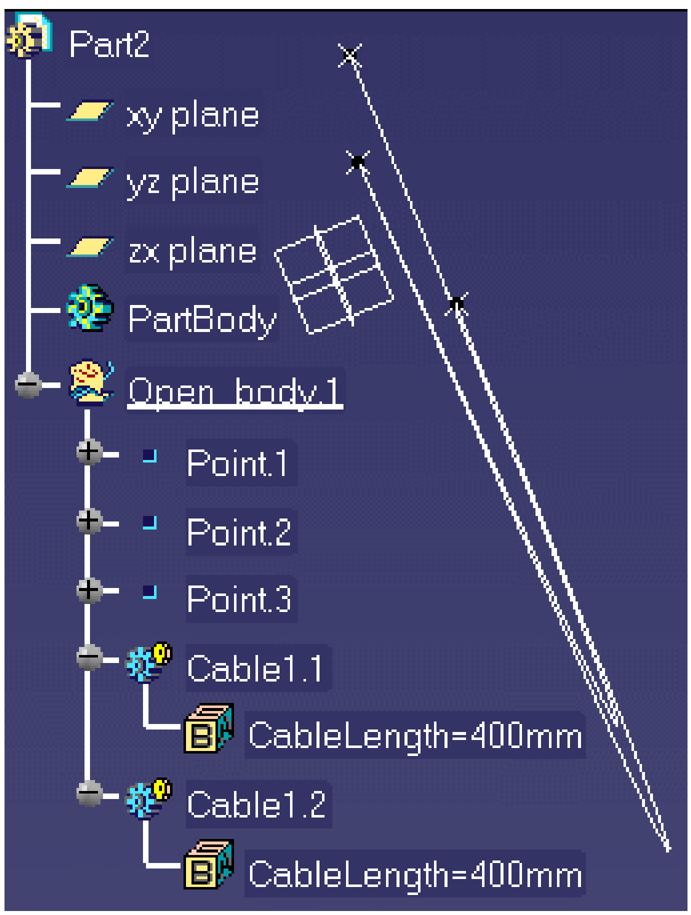
14. Save the file and close it.

15. Create a new .CATPart file, access the Generative Shape Design workbench, and create 3 points.

16. Select the **Insert->Instantiate From Document...** command. The File Selection panel opens. Select the KwrEvent_BeforeUpdate.CATPart file you just saved and click the **Open** button.

17. The Insert object dialog box opens. Select Point.1 and Point.2 in the geometry or in the specification tree and click **OK**. The cable (UDF) is instantiated and the optimization is launched before the update.

18. Repeat steps 14 and 15: select Point.2 and Point.3 when instantiating the UDF: the cable lengths are optimized.



19. Double-click the **CableLength=400mm** parameter and change its value to 200mm. This cable length is optimized once again just before the update.

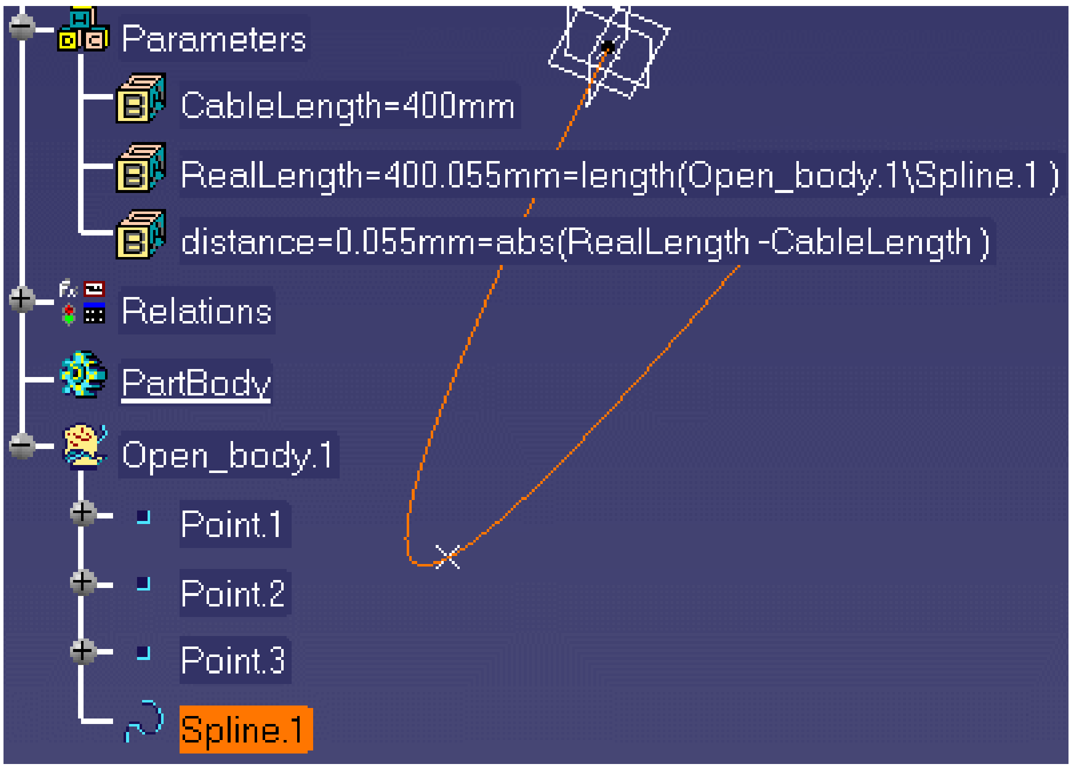
To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Reaction: ValueChange Event

This task explains how to use the ValueChange event associated to the reaction feature.
The CATPart file contains a cable going through 3 points. The user wants the cable length to be optimized each time he modifies the cable length.
The scenario is divided into 3 parts:

- the user creates an optimization
- the user creates a reaction
- the user modifies the cable length value

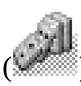
1. Open the KwrEventValueChange.CATPart file: It contains 3 points and a spline (called cable in this scenario).

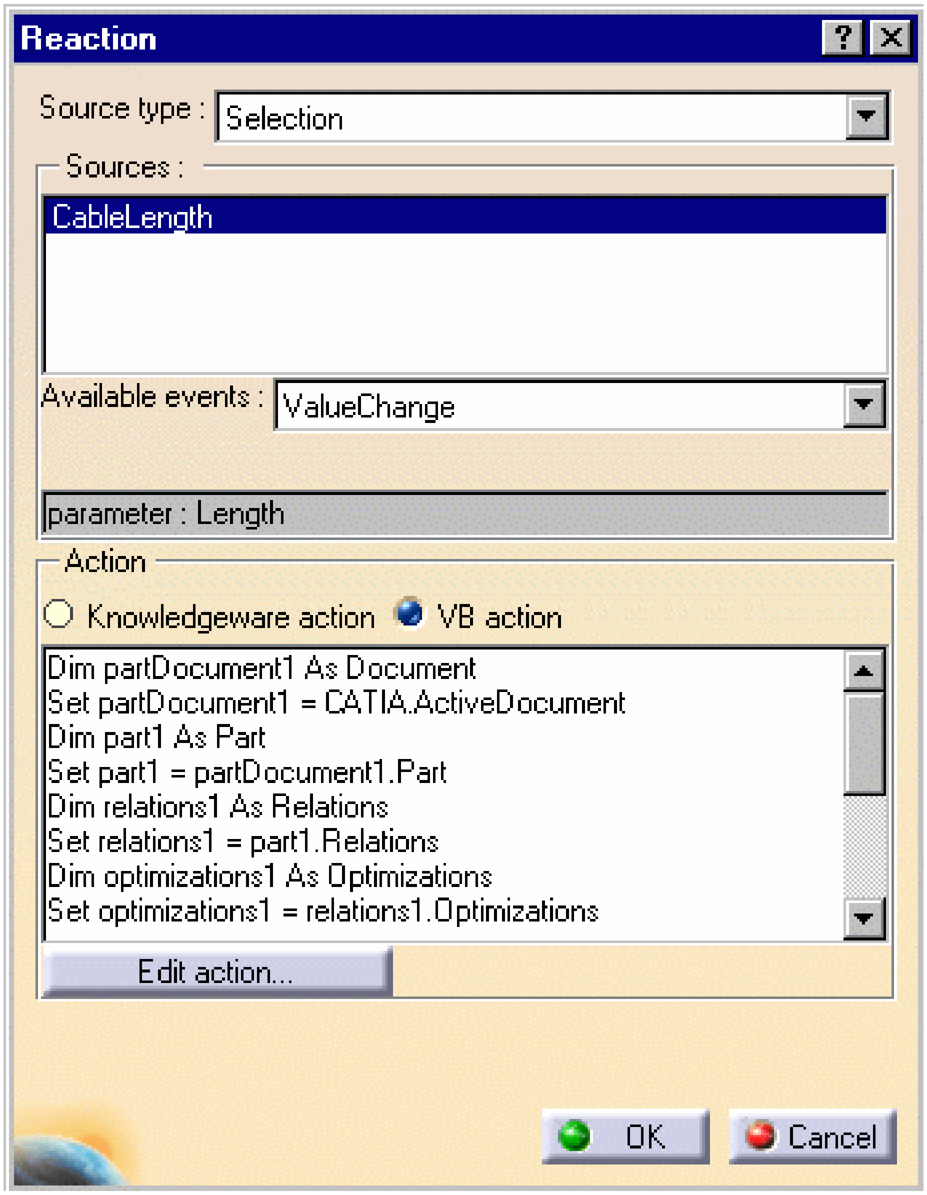


2. From the **Start->Knowledgeware** menu, access the Product Engineering Optimizer workbench and click the **Optimize** icon ( ). The Optimization window opens.
3. Enter the following data in the Optimization window:

Problem tab		
	Optimization type	**Minimization**

	Optimized parameter	**distance**
	Free parameters	**Geometrical Set.1\Point.3\Z**
	Algorithm	**Simulated Annealing-Convergence speed**
	Termination criteria	Maximum number of updates: **100** Consecutive updates without improvements: **20** Maximum time (minutes): **5**
Constraints tab		
	New constraint	**`Geometrical Set.1\Point.3\Z` - max (Geometrical Set.1\Point.2\Z ,Geometrical Set.1\Point.1\Z) <= 0mm**

4. Click **OK** in the opening dialog box, click **Run optimization.**
5. Select an output file and click **Save**.
6. Click **OK** once the optimization process is over.
7. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench and click the Reaction icon (). The Reaction dialog box opens.



- In the **Source type** field, select **Selection** and select the CableLength parameter in the specification tree for the Reaction to be applied to the CableLength parameter.

- In the **Available events** list, select **ValueChange**.

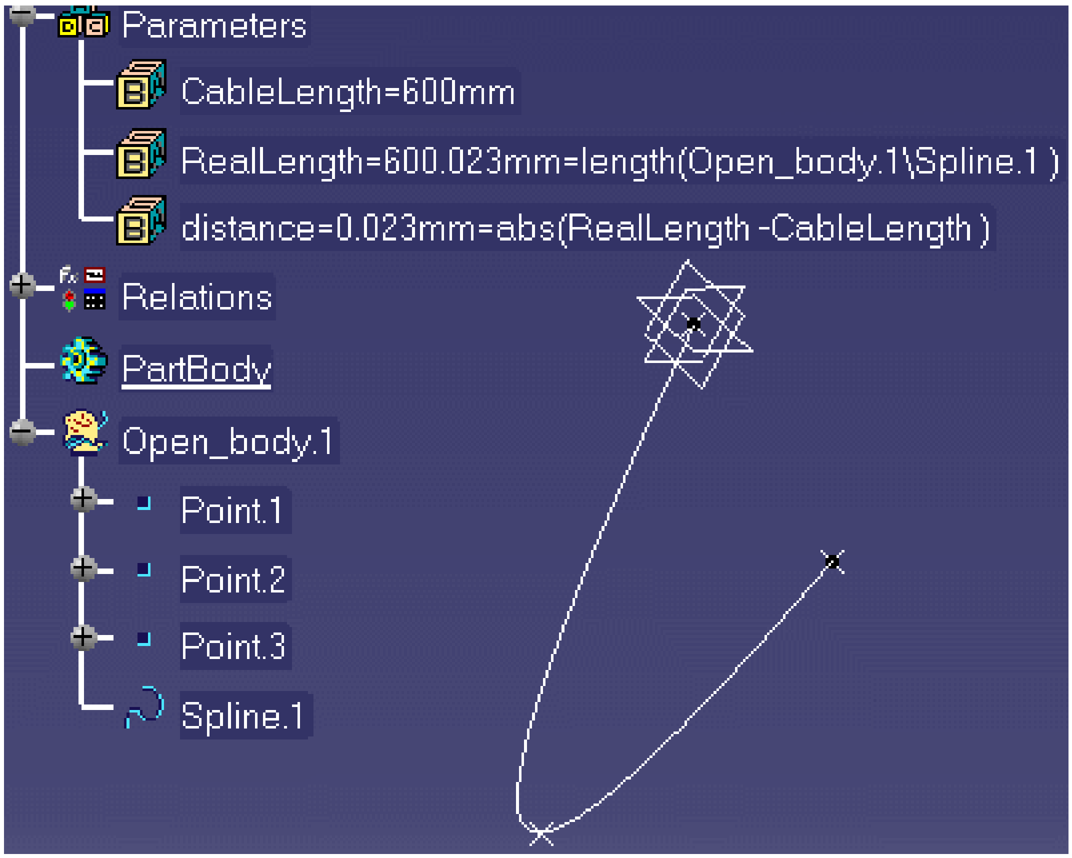
- In the **Action** field, select **VB action**.

8. Click the **Edit action...** button, paste the following script in the editor, and click **OK** twice:

```
Dim partDocument1 As Document
Set partDocument1 = CATIA.ActiveDocument
Dim part1 As Part
Set part1 = partDocument1.Part
Dim relations1 As Relations
Set relations1 = part1.Relations
Dim optimizations1 As Optimizations
Set optimizations1 = relations1.Optimizations
Dim anyObject1 As Optimization
Set anyObject1 = optimizations1.Item("Optimization.1")

anyObject1.Run False
```

The reaction is added to the specification tree.

9. Double-click twice the **CableLength=400mm** parameter and change its value to 600mm: The optimization is launched (the RealLength and the distance parameters have changed) and the geometry is changed accordingly.



To know more about the Reaction feature window, see Using the Reaction Feature Window.

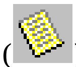
Using a Reaction with a User Feature: Instantiation Event

This task explains how to use a reaction in a User Defined Feature.
The scenario described below is divided into two major steps:

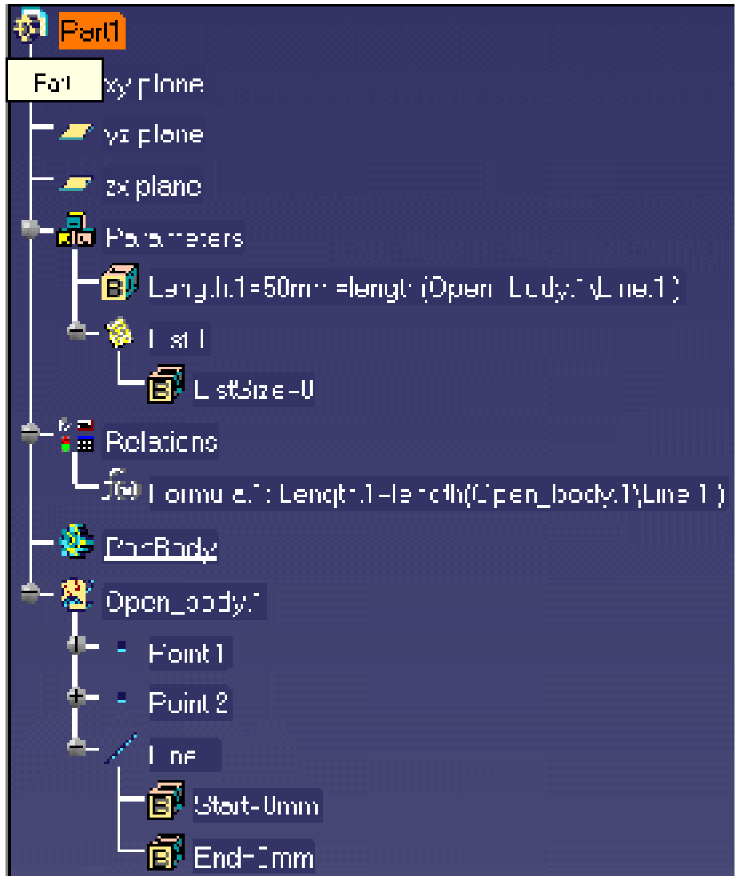
- In the first step, you first create a formula that returns the length of the line, you create a reaction that will add items of length type to a list when the document is instantiated and then you create a UDF containing the line, the reaction and the formula.
- In the second step, you open a second document, you create a rule based on a list that will display the total length, and then you instantiate the UDF that you previously created in this document.

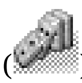
A basic understanding of the Part Design workbench and of Product Knowledge Template is required to carry out this scenario.

1. Open the KwrUDFandReaction.CATPart file.
2. Create a parameter of **Length** type and assign it a formula. To do so, proceed as follows:
   - Click the icon, select **Length** in the **New Parameter of Type** scrolling list, click the **New Parameter of type** button, and click the **Add Formula** button.
   - In the **Dictionary**, click **Measures**, and double-click **length (Curve,): Length**.
   - Position the cursor between the parentheses and double-click Line.1 in the specification tree. Click **OK**, **Yes**, and **OK**. (Length.1=length(Geometrical Set.1\Line.1)).

3. Access the Knowledge Advisor workbench, click the List icon () to create a list, and click **OK**. An empty list appears under the Parameters node.

(Click the graphic opposite to enlarge it.)



4. Click the Reaction () icon. The Reaction editor opens:

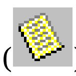
   - In the **Source type** list, select **Owner**.
   - In the **Available events**, select **Instantiation.**
   - In the **Action** area, select **Knowledgeware action** and click the **Edit action** button. The Action editor opens.
   - Click the list in the specification tree and, in the Dictionary pane select List, and in the Member pane, double-click **List.AddItem**.

- Position the cursor between the parentheses and enter Length.1 before the comma, and 0 after the coma. Click **OK** twice. The Reaction feature is created.

5. Access the Part Design workbench and select the **Insert->UserFeature->UserFeature creation** command. The Userfeature Definition window opens: in the **Definition** tab, enter the name of the User Feature (UserFeature1 in this scenario) and select the Line, the Reaction, and the Length parameter in the specification tree. Click **OK**.

   The UserFeature1 is created and displayed under the Knowledge Templates node.

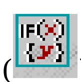
6. Save your file, close it, and open the KwrUDFandReaction2.CATPart file This is the file into which you will instantiate the UDF you previously created.

7. Access the Knowledge Advisor workbench, and click the list icon () to create an empty list and click **OK**.

8. Create a parameter of **Length** type (called Length.1 in this scenario) and apply a formula to it. To do so, proceed as follows:

   - Click the  icon, select **Length** in the scrolling list, click the **New Parameter of type** button, and click the **Add Formula** button.

   - Select the list in the specification tree, in the Dictionary pane select List, and in the Member pane, double-click **List.sum**. Click **OK** three times.

9. Click the Rule icon (), click **OK**, enter the following script in the Rule Editor, and click **OK**:

```
Message("Total Length : #",Length.1 )
```

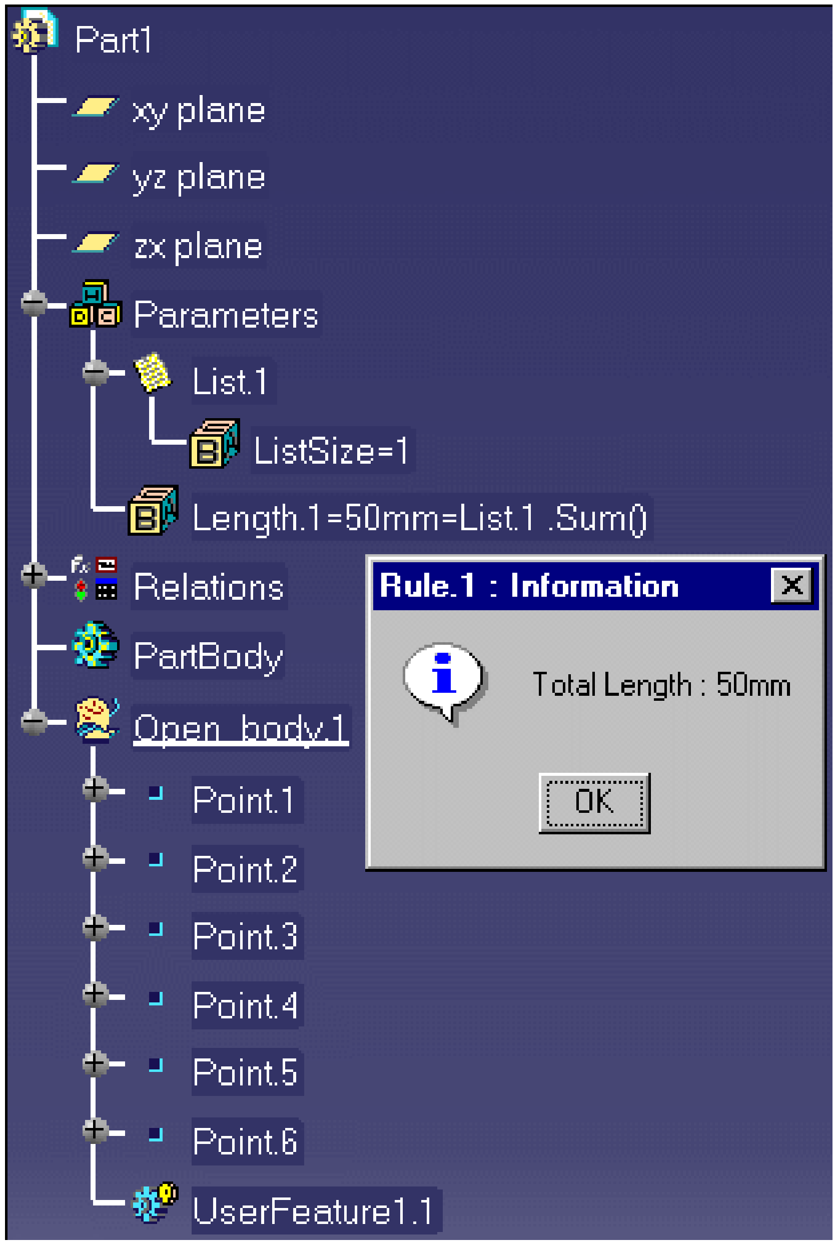
   The total length is displayed: 0mm.

10. Access the Part Design workbench and select the **Insert->Instantiate from Document** command.

11. Select the file you created (from step 1 to step 5, KwrUDFandReaction.CATPart in this scenario) and click **Open**. The **Insert Object** window opens:

- Select Point.1 in the specification tree or in the geometry.
- Select Point.2 in the specification tree or in the geometry.
- Select List.1 in the specification tree and click **OK**.

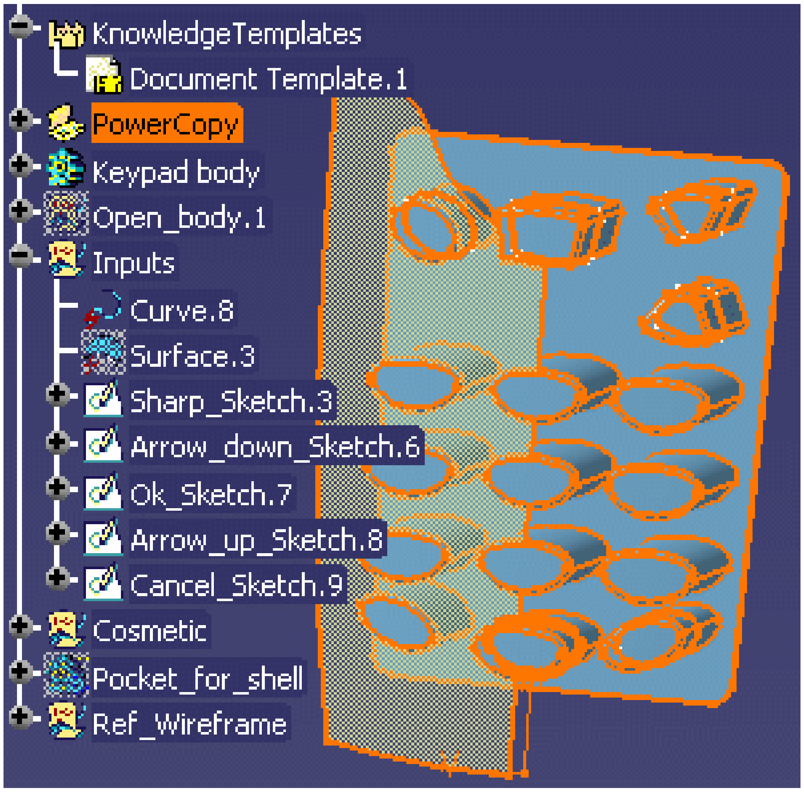
The rule is fired and the Total Length is displayed.



Using a Reaction with a Document Template: Instantiation Event

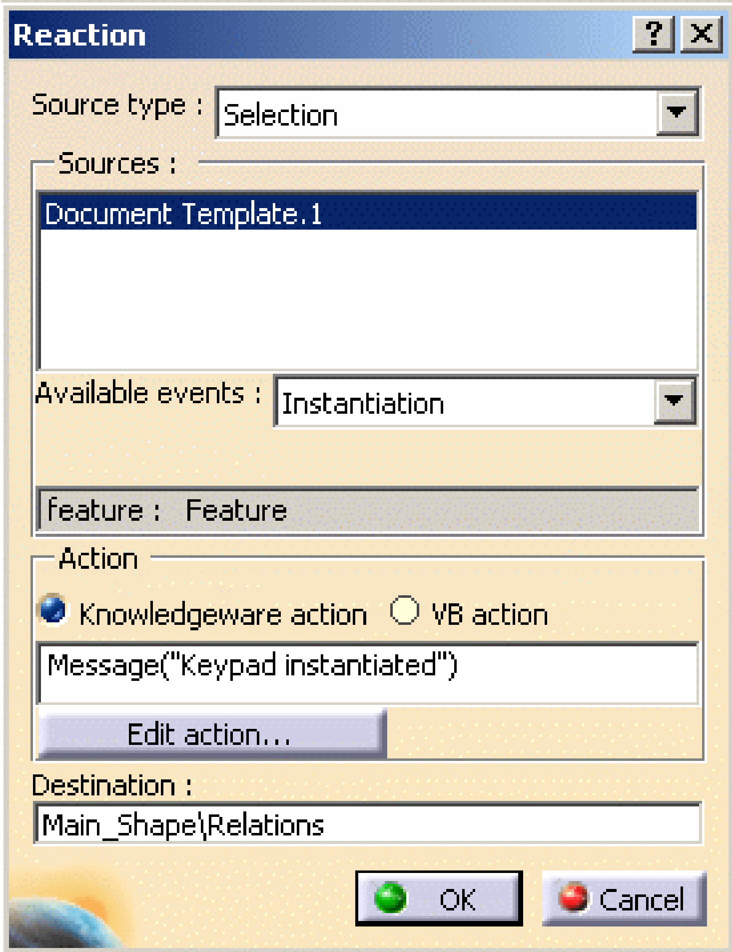
This task explains how to use the Instantiation event associated to the reaction feature.
The user wants to instantiate a document template containing a keypad into a .CATProduct file already containing a mobile phone support.

1. Open the KwrInstantiationEvent.CATPart file. The following image displays.



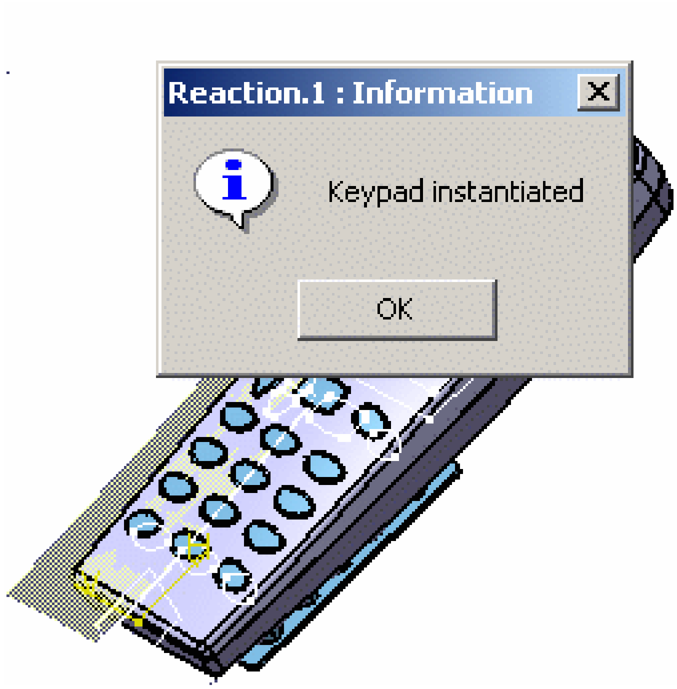
2. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench and click the Reaction icon ( ). The Reaction dialog box opens.

- In the **Source type** field, select **Selection** and select Document Template.1 below the Knowledge Templates node.



- In the **Available events** list, select **Instantiation**.
- In the **Action** field, select **Knowledgeware action** and enter the following message: Message("Keypad instantiated"). Click **OK** when done. The Reaction feature is added to the Relations node.

3. Click **OK** when done. The reaction is added to the specification tree.
4. Save your file and close it.
5. Open the KwrInstantiationEvent.CATProduct file.
6. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench.
7. Click the **Instantiate From Document...** icon.
8. In the **File Selection** dialog box, select the KwrInstantiationEvent.CATPart file that you have just saved and click **Open**. The Insert Object dialog box displays.
9. In the Insert Object dialog box, click the **Use identical name** button.
10. Make the appropriate selection in the Replace Viewer window and click **Close** when done. Click **OK** in the Insert Object dialog box. The document template is instantiated and the reaction is launched.

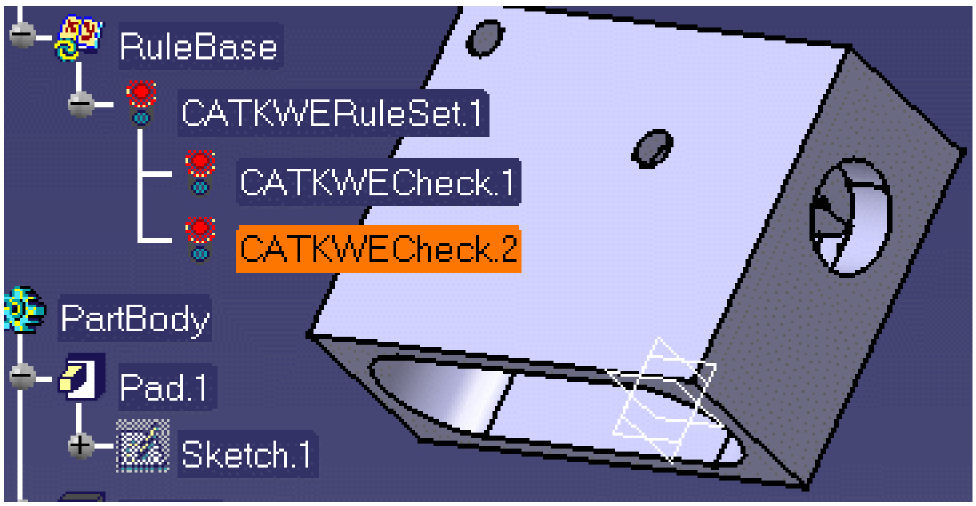


To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Reaction: Update Event

This task explains how to use the Update event associated to the reaction feature.
The CATPart file contains a rulebase that is updated each time a modification is made.

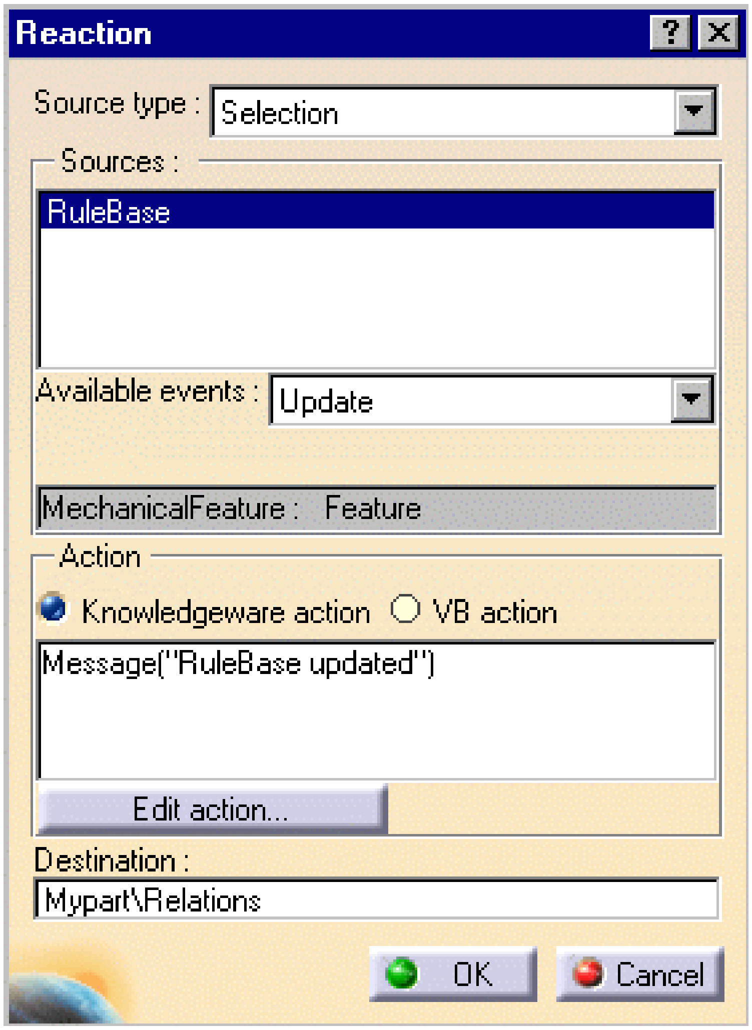
1. Open the KwrEvent_Update.CATPart: It contains a part with holes and a rulebase made up of 2 checks.



2. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench and click the Reaction icon ( ). The Reaction dialog box opens.

- In the **Source type** field, select **Selection** and select the RuleBase in the specification tree for the Reaction to be applied to the rulebase.

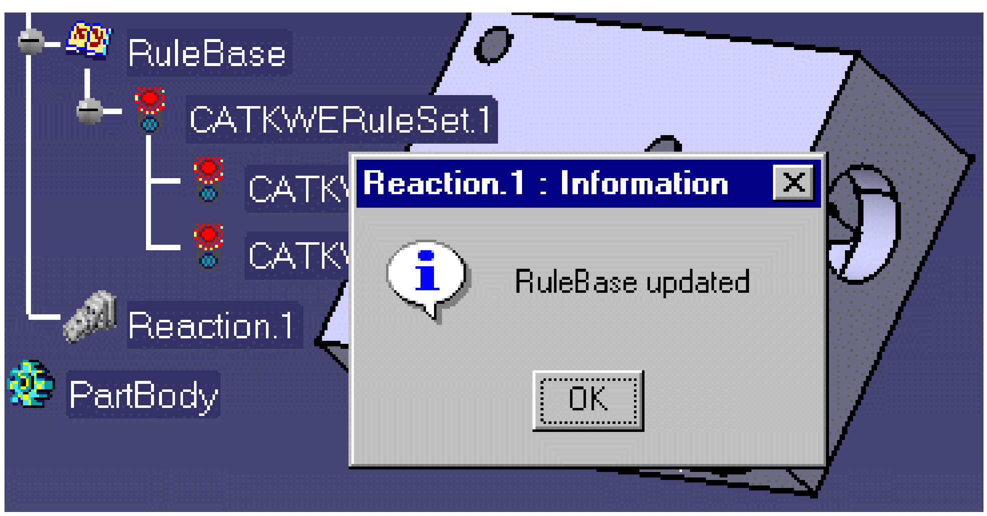
- In the **Available events** list, select **Update**.



- In the **Action** field, select **Knowledgeware action** and enter the following message: Message("Rulebase updated").

3. Double-click the CATKWECheck.1. The Check Editor opens. Modify the check: (H\Diameter == 20mm) and click **OK**

4. Right-click the rulebase and select the **Rulebase object->Manual Complete Solve** command. The reaction is fired and the following message displays:

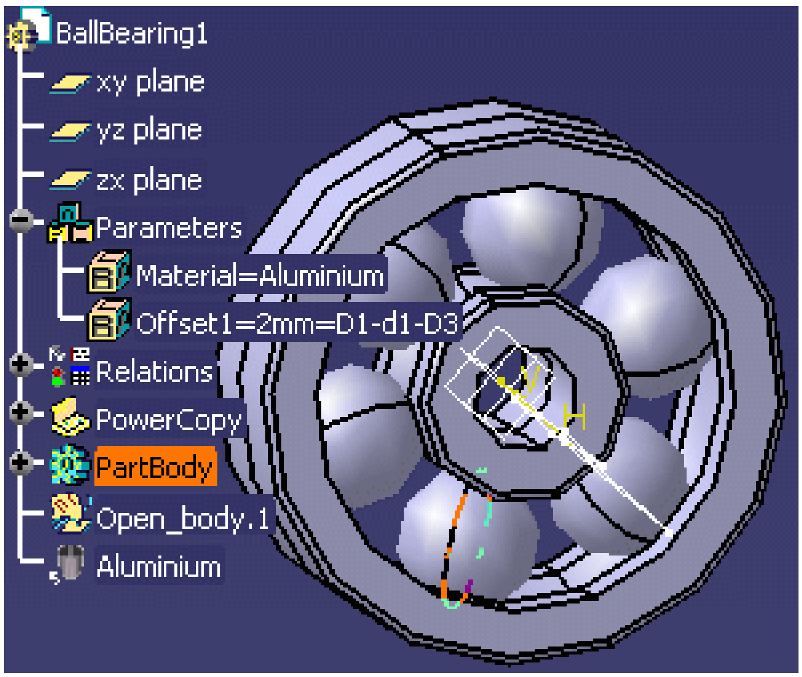


To know more about the Reaction feature window, see Using the Reaction Feature Window.

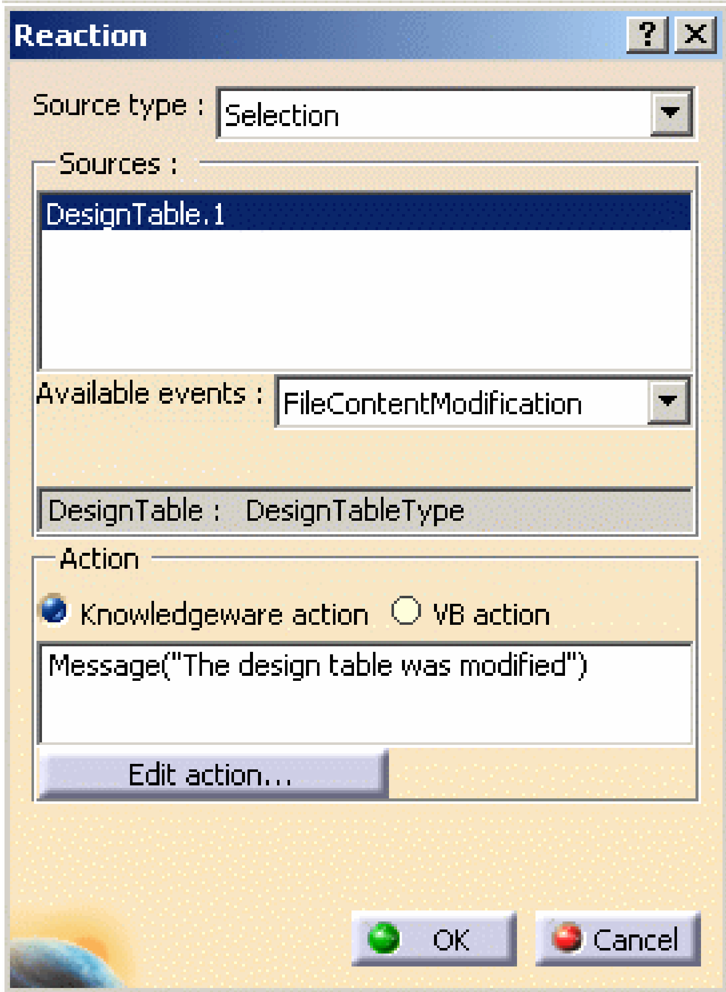
Creating a Reaction: File Content Modification Event

This task explains how to use the FileContentModification event associated to the reaction feature. This event launches a reaction each time the file associated to the design table is modified.

1. Open the KwrBallBearing1.CATPart file. The following picture displays.



2. Click the **Design Table** icon ( ). The Creation of a Design Table dialog box displays.
3. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.
4. Select the KwrBearingDesignTable.xls file and click **Open**. Click **Yes** when asked if you want to associate the columns of the tables with the parameters.
5. Click **OK** to apply the default configuration.
6. From the **Start->Knowledgeware** menu, access the Knowledge Advisor workbench and click the **Reaction** icon ( ). The Reaction dialog box opens.



- In the **Source type** field, select **Selection** and select the DesignTable.1 in the specification tree for the Reaction to be applied to the design table.
- In the **Available events** list, select **FileContentModification**.
- In the **Action** field, select **Knowledgeware action** and enter the following message: Message("The design table was modified").
  Click **OK** when done. The Reaction feature is added to the Relations node.

7. Double-click DesignTable.1 in the specification tree. The Design Table window displays.
8. Click the **Edit table...** button and change the material of row 2 to Gold. Save your file and close it. The reaction is launched and the message displays.

To know more about the Reaction feature window, see Using the Reaction Feature Window.

Creating a Loop in a Reaction

This task explains how to create a loop in a reaction. In the scenario described below, the user:

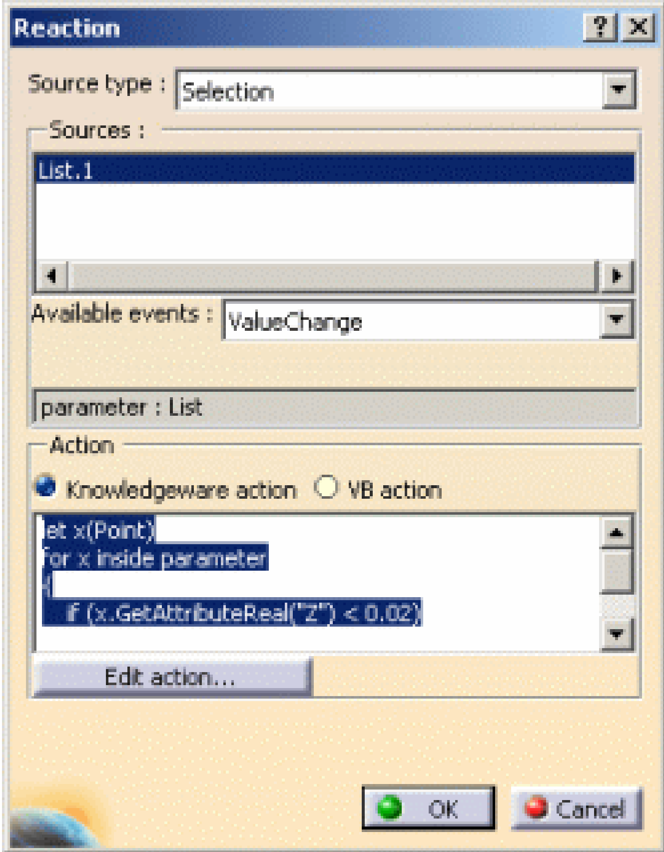
- Creates 3 points that are inserted into a list
- Creates a formula that will automatically fill in the list
- Creates a reaction that will set the Z coordinate of the points to 20mm if this value is inferior to 20mm.
- Creates a reaction that will set the Y coordinate of the points to 40mm if this value is inferior to 40mm.
- Updates the Part

It is now possible for users to create the While and the For constructs. To know more, see Using the Knowledge Advisor Language.

1. Create a Part containing 3 points.
2. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench.
3. Click the **List** icon ( ) to create the list that will contain the created points.
4. Add a formula to fill in the list. To do so, proceed as follows:
   - Click the Formula Editor icon. The Formula Editor displays. In the specification tree, select List.1 and click the **Add formula** button.
   - List.1=PartBody.Query("Point",""). Click **OK** twice when done.

6. Click the **Reaction** icon ( ). The Reaction dialog box opens.

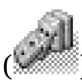
- In the **Source type** field, select **Selection** and select the list in the specification tree for the Reaction to be applied when the list is modified.
- In the Select a source... dialog box, select List.1 and click **OK**.

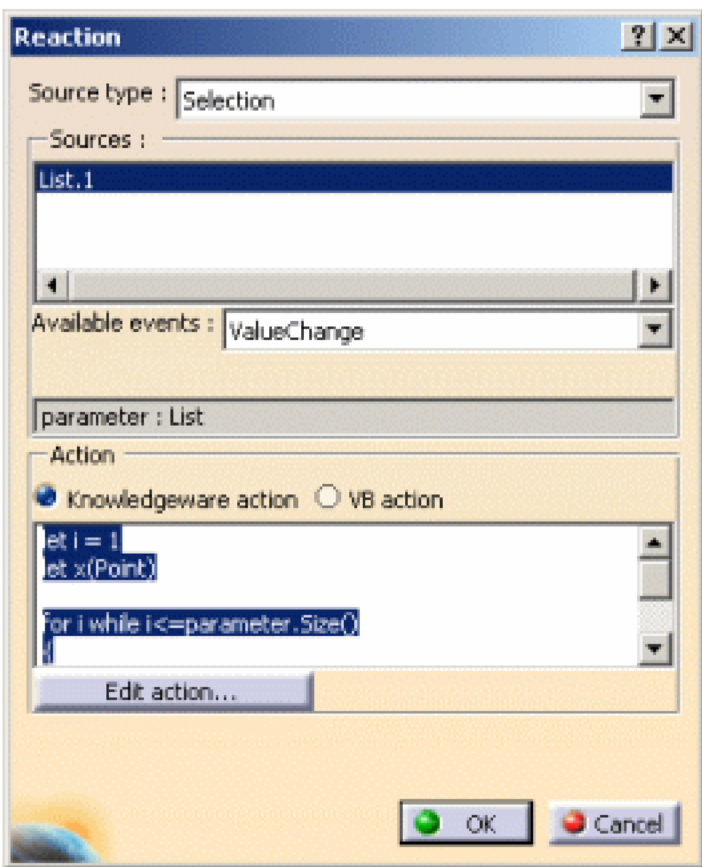


- In the **Available events** list, select **ValueChange**.
- In the **Action** field, select **Knowledgeware action** and enter the following message action body.

```
let x(Point)
for x inside parameter
{
if (x.GetAttributeReal("Z") <
0.02)
x.SetAttributeReal("Z",0.02)
}
```

- Click **OK** when done. The Reaction feature is added to the Relations node.

7. Click the **Reaction** icon (). The Reaction dialog box opens.



- In the **Source type** field, select **Selection** and select the list in the specification tree for the Reaction to be applied when the list is modified.
- In the Select a source... dialog box, select List.1 and click **OK**.
- In the **Available events** list, select **ValueChange**.

In the **Action** field, select **Knowledgeware action** and enter the following message action body.

```
let i = 1
let x(Point)

for i while i<=parameter.Size()
{
x = parameter.GetItem(i)
if (x.GetAttributeReal("Y") <
0.04)
x.SetAttributeReal("Y",0.04)
}
```

- Click **OK** when done. The Reaction feature is added to the Relations node.

8. Create a new point with the following coordinates:

   X: 36mm

   Y: 12mm

   Z: 0mm

9. Update the Part twice: The Y and Z coordinates are set to 40 and 20 mm. Click here to displays the result sample.

To know more about the Reaction feature window, see Using the Reaction Feature Window.

Launching a VB Macro with Arguments

The task below illustrates how to add arguments to a macro.

Macros with arguments are features that can be:

- stored in CATPart or CATProduct documents,
- stored in catalogs. Double-click them in the catalog to run them,
- called from a rule (VBScriptRun) or a reaction. In this case, arguments are passed from the rule.

The icon enables you to access the macro editor. In addition to the usual 'edit and run' capabilities, this editor allows you to:

- specify arguments
- carry forward a feature definition to the editor just by selecting the feature either in the tree or in the geometry area.

Note that the VB macro with arguments functionality is intended to help users write macros. It is not intended to find the code of other objects: Only the objects that were correctly integrated to the macros are supported.

1. Click the icon. The script editor is displayed.
2. Copy/paste the script below into the editor:

```
Dim oActiveDoc As Document
Set oActiveDoc = CATIA.ActiveDocument

If (InStr(oActiveDoc.Name,".CATPart")) <> 0 Then

Dim oParams As Parameters
Set oParams = oActiveDoc.Part.Parameters

Dim strParam1 As StrParam
Set strParam1 = oParams.CreateString("FirstName", "")

Dim strParam2 As StrParam
Set strParam2 = oParams.CreateString("LastName", "")

strParam1.Value = fname
strParam2.value = lname

Else MsgBox "The active document must be a CATPart"
End If

End Sub
```

3. Enter the fname and lname arguments in the field located between the parentheses. The arguments must always be separated by a comma.

4. Click OK to add the macro to the document. A 'VB Scripts' sub-node is added to the specification tree below the Relations node. A VB Script object is added below this sub node.

5. Double-click the VB Script object. The Script Editor is displayed. The Insert Object Resolution button allows you to retrieve a feature definition. The VB Script.2 macro of the KwrObject.CATPart sample illustrates how to use this capability.

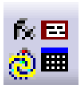
6. Click **Run script...**. The Select Inputs for Script Arguments is displayed.

7. If need be, select fname in the argument list, then enter a string into the value field (no quotation marks). Then select lname and enter the lname value.

8. Click OK to run the script. The two string type parameters are added to the document. Their values are those you have just specified.

Note that the VB script features with arguments are provided with a contextual menu enabling the user to launch the script.

Working with Relations

 Select the Add Set of Relations icon to regroup relations into categories.

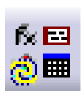
 Select the Measure Update icon to perform the update of relations using measures.

Creating Sets of Relations

This task explains how to create sets of relations below the Relations node of the specification tree.

Using this capability enables you to regroup relations into categories. When you create a relation, you are prompted to enter a *destination*. i.e. a feature you add the new relation to.

Formulas, design tables, rules and checks can all be created into relation sets. When no relation set has been created, the destination field of the relation editor is by default initialized to the Relations node.

1. Create a Part and from the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench.
2. Click the  icon and click the Relations node, the Relations.1 (or Relations.*n*) relation set is added to the specification tree right below the Relations node.
3. Click the Rule or the Check creation icon. The first dialog box displayed is similar to the one displayed when you create a relation right below the Relations node except that you must specify a destination.
   To do so, select the value specified in the **Destination** field of the relation editor, then select the Relation Set you want to add a relation to. This results in a modification of the destination path in the relation editor (*partname*\Relations.*n* is replaced with *partname*\Relations\Relations.*n*).
4. Click **OK** to display the next dialog box and enter the relation body.
5. After you have finished specifying the new relation, click **OK** in the editor dialog box. In the specification tree, you can expand the feature which represents the relation set. A new relation has been added below this relation set.

Using Relations based on Publications at the Product Level

This scenario explains how to use relations based on publications at the product level. The scenario described below is divided into the following steps:

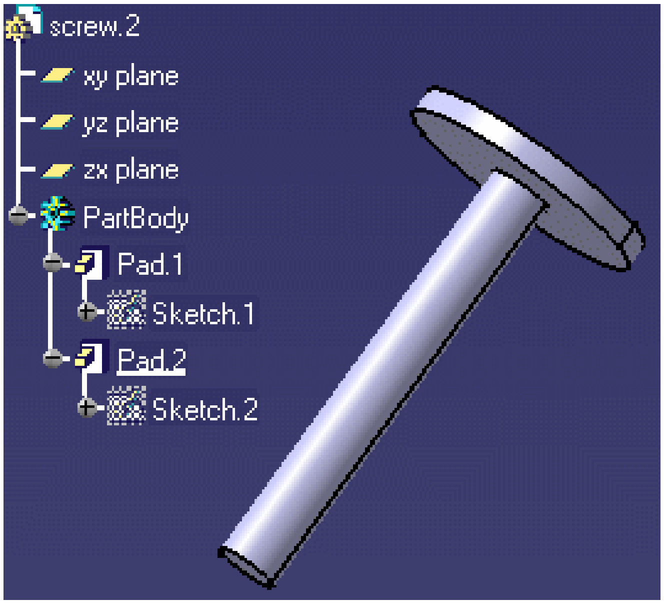
- Add a parameter to the KwrScrew.CATPart called Screw_Volume, add a formula to calculate the volume of the screw and publish the Screw_Volume parameter.
- Add a parameter to the KwrScrew1.CATPart called Screw_Volume, add a formula to calculate the volume of the screw and publish the Screw_Volume parameter.
- Create a CATProduct file called Bolt and import the KwrScrew.CATPart
- Import KwrNut.CATPart.
- In the context of the Bolt product, create a formula calculating the bolt volume based on the screw and the nut publications.
- In the context of the bolt, replace KwrScrew.CATPart by KwrScrew1.CATPart. The volume is recomputed.

Before you start, make sure that the **Keep link with selected object** check box is checked (**Tools->Options->Infrastructure->Part Infrastructure->General**).

Note that this function can be used with:

- Design Tables
- Formulas
- Rules and Checks
- Set of Equations
- The optimization

1. Open the KwrNewScrew.CATPart document. The following image displays.



2. Add a Volume parameter to the part. To do so, proceed as follows:

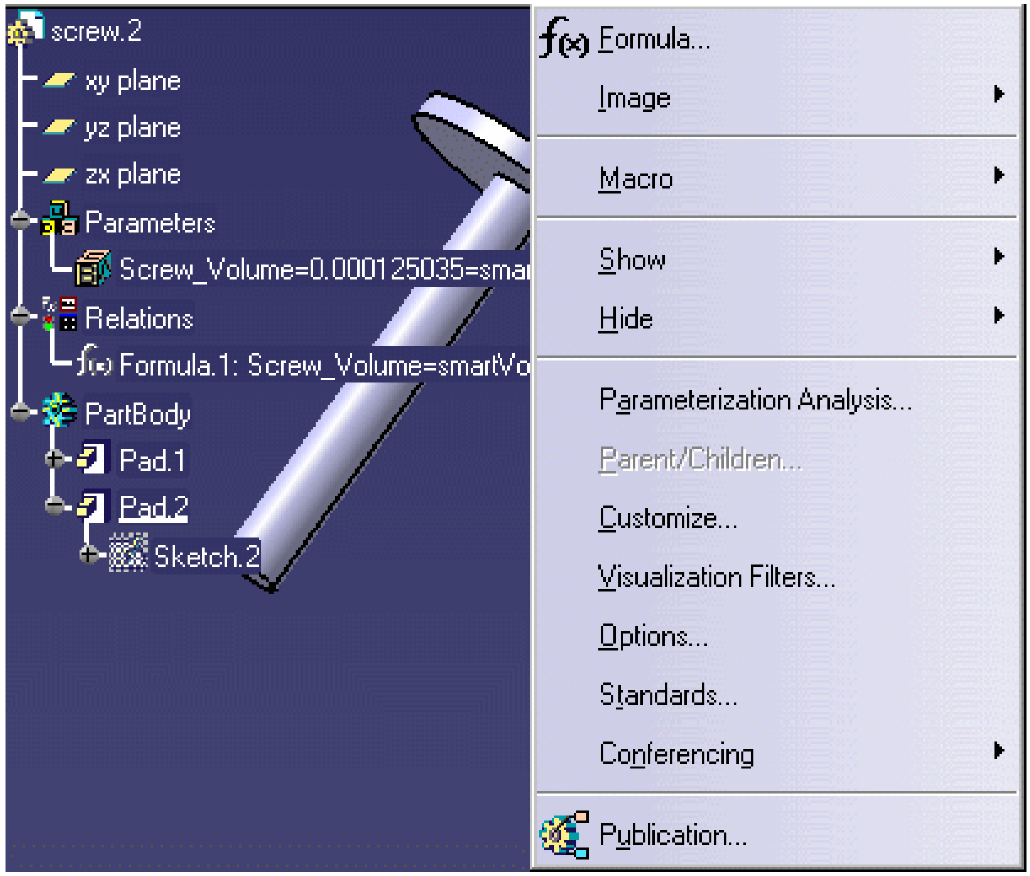
- Click the  icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Real** and click the **New parameter of type** button.

- In the **Edit name or value of the current parameter** field, enter the name of the parameter: **Screw_Volume**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.

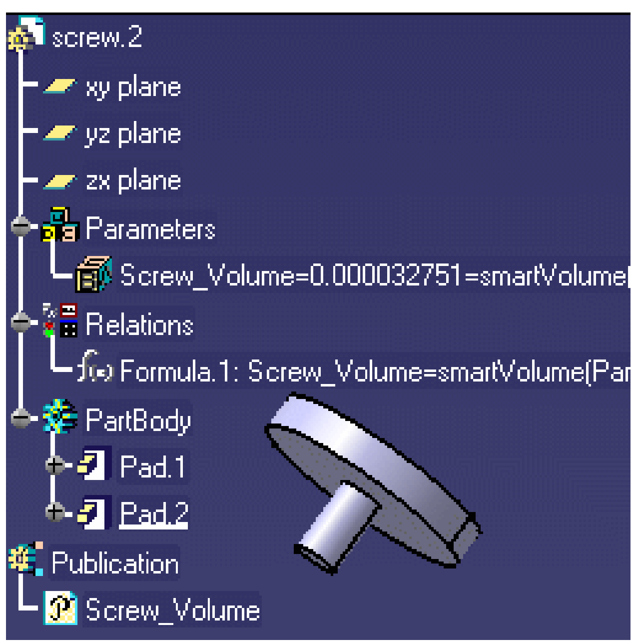
- Enter the following formula by using the Dictionary: **Screw_Volume=smartVolume(PartBody\Pad.1)+smartVolume(PartBody\Pad.2)**. Click **OK** three times.

3. Publish the **Screw_Volume** parameter.

To do so, select the **Tools->Publication** command and click the **Screw_Volume** parameter under the Parameters node in the specification tree. Click **OK**. The published parameter appears in the specifications tree below the Publication node. Save your file and close it.



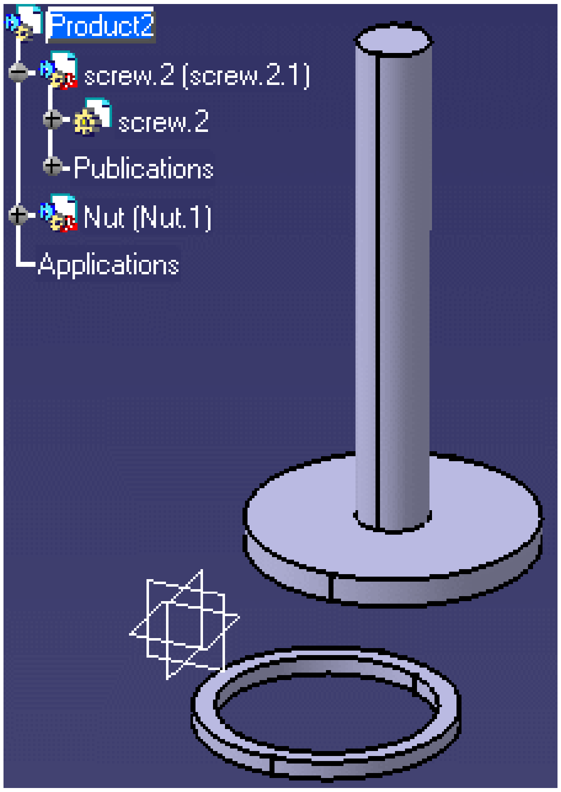
4. Open the KwrNewScrew1.CATPart and repeat the steps listed above (steps 1 to 3 included). The part should be identical to the one below. Save your file and close it.



5. Create a CATProduct file named KwrBolt.CATProduct.

6. Click the Root product and select the **Insert->Existing Component...** command. The **File selection** box displays. Select the KwrNewScrew.CATPart file and click **Open**. The screw is imported.

7. Select the **Insert->Existing Component...** command, select the KwrNewnut.CATPart file and click **Open**. The nut part is inserted.

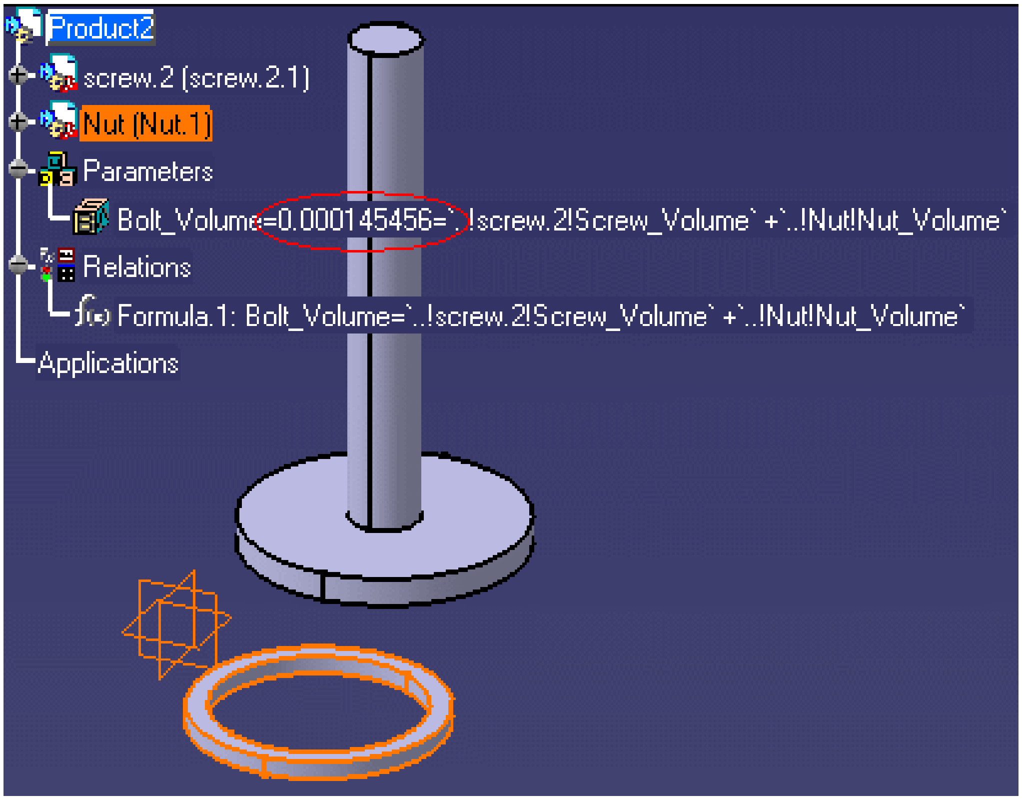


8. Add a Bolt_Volume parameter to the product to compute the volume of the bolt. To do so, proceed as follows:

- Click the Root product and click the  icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Real** and click the **New parameter of type** button.

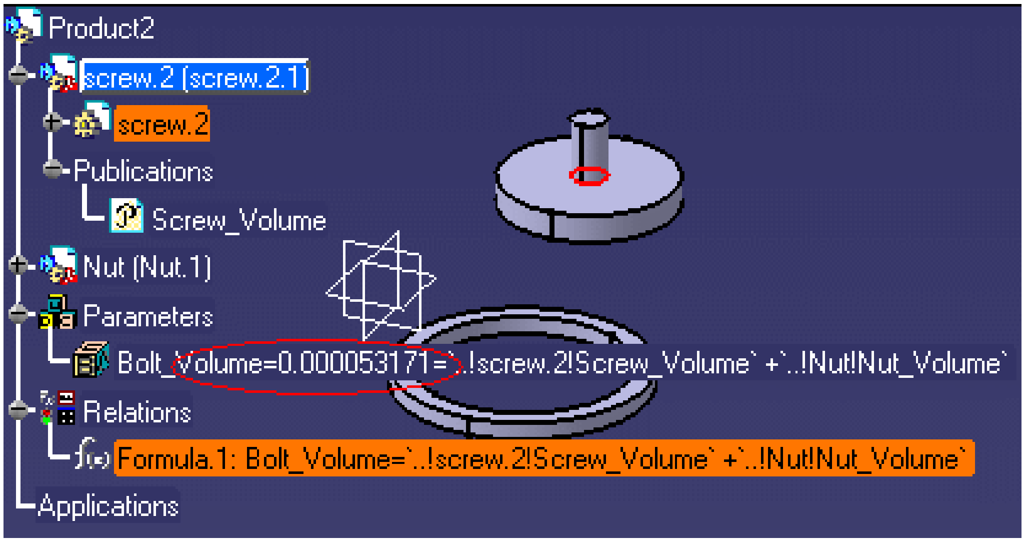
- In the **Edit name or value of the current parameter** field, enter the name of the parameter: **Bolt_Volume**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.

- Enter the following formula by using the Dictionary and by clicking the published parameters in the specification tree: **Bolt_Volume=`..!screw.2!Screw_Volume` +`..!Nut!Nut_Volume`**. Click **OK**, and **OK**. The Bolt volume displays



9. Replace the screw to compute a new volume: Double-click, then right-click the Screw.2 component in the specification tree and select the **Components->Replace Component...** command. The **File Selection** window opens. Select the KwrNewScrew1.CATPart file and click **Open**.

10. Click **Yes** and **OK** in the Impacts on Replace window. The new screw is inserted and the bolt volume is updated.


Product2
screw.2 (screw.2.1)
screw.2
Publications
Screw_Volume
Nut (Nut.1)
Parameters
Bolt_Volume=0.000053171=`..!screw.2!Screw_Volume` +`..!Nut!Nut_Volume`
Relations
Formula.1: Bolt_Volume=`..!screw.2!Screw_Volume` +`..!Nut!Nut_Volume`
Applications

Activating and Deactivating a Component

This task explains how to activate and deactivate a component.

In the scenario described below, the CATProduct file contains two CATPart files that you will activate and deactivate alternatively after creating user parameters and a rule based on these parameters.

Parameters driven by rules are designed to enable the user to control components activities at assembly level.

When working in a Japanese environment, remember to check the **Surrounded by the Symbol'** option (**Tools->Options->General->Parameters and Measure->Knowledge** tab).

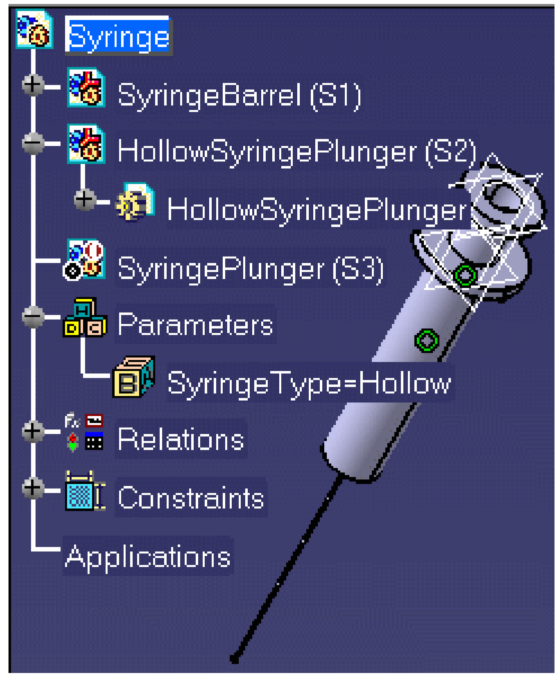
1. Open the KwrSyringe.CATProduct file and save the following files in the same directory (SyringePiston.CATPart, HollowSyringePiston.CATPart, and SyringeContainer.CATPart): This file contains a syringe made up of three different parts: A barrel, and two different plungers.
2. Create a multiple value parameter of string type. To do so, proceed as follows:

- Click the f(x) icon. The Formulas Editor opens.
- Select String in the scrolling list with Multiple Values. Click the **New Parameter of type** button. The **Value List** dialog box opens.
- Enter two different values, Hollow and Full, and click **OK**.
- Edit the name of the new parameter (SyringeType in this scenario) in the **Edit Name or value of the current parameter** and click **OK**. The new parameter is displayed under the **Parameters** node of the Specification tree.

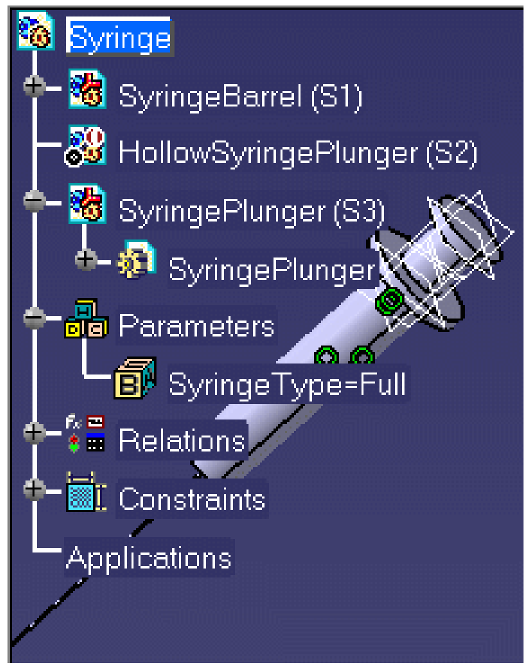
3. Access the Knowledge Advisor workbench and click the Rule icon to create a rule. The script of this rule will allow you to enable or disable one of the plungers.
4. Enter the code below in the Rule Editor, and click **OK**.

```
if (SyringeType == "Hollow")
{
S3\Component_Activation_State = false
S2\Component_Activation_State = true
}
else
{
S2\Component_Activation_State = false
S3\Component_Activation_State = true
}
```

6. Double-click the SyringeType parameter under the Parameters node and select Hollow in the Edit Parameter window. The SyringeBarrel CATPart and the HollowSyringePlunger CATPart are displayed.



7. Double-click the SyringeType parameter and select "Full" in the Edit Parameter window. The SyringeBarrel CATPart and the SyringePlunger CATPart are displayed.

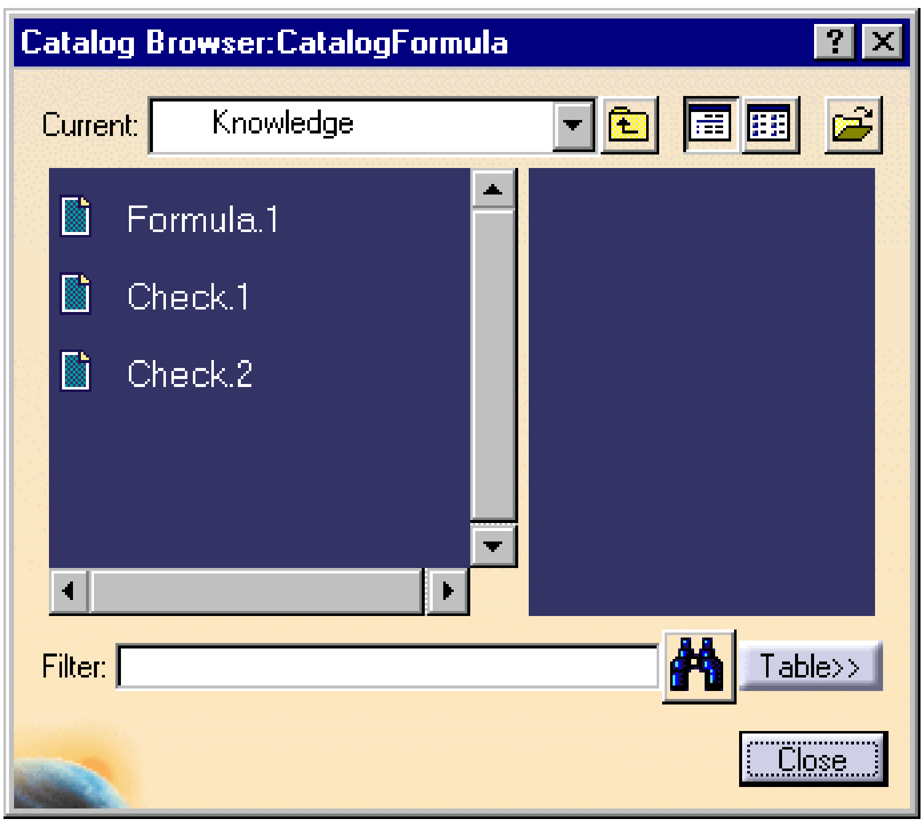


Instantiating Relations from a Catalog

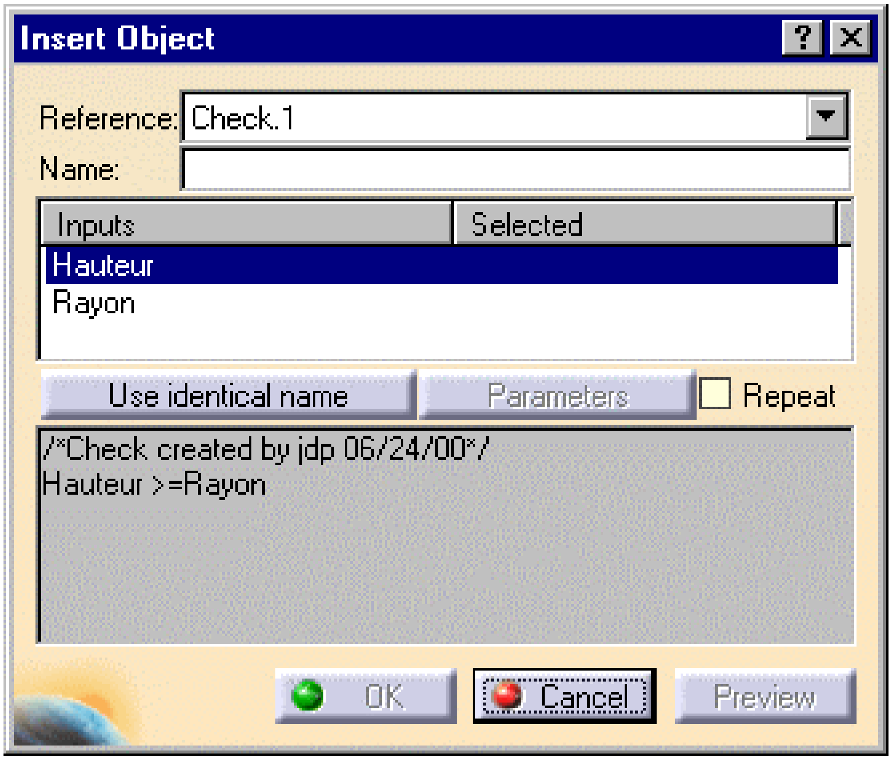
The scenario developed below explains how to instantiate a check stored in a catalog into a CATPart document.

Formulas, rules and checks can be stored in a catalog. They can then be reused in a document by using an instantiation mechanism. For more information about catalogs, see the *Infrastructure User's Guide*.

1. Open the Formula_005_Start.CATPart file.
2. In the Tools toolbar, click the icon. The catalog browser is displayed.
3. Click the icon to open the CatalogFormula.catalog catalog. The catalog browser looks something like the one below (you may need to expand the Knowledge node to display the three relations Formula.1, Check.1, Check.2 - what you see in the left-hand part of the Catalog Browser depends on the last interactions you have carried out with this dialog box).



4. Double-click the Check.1 object. The dialog box below is displayed.



5. Rename the Check.1 check by using the Name field. Enter HeightCheck for example.

6. The 'Hauteur' input is highlighted. In the part specification tree, select the "Hauteur" parameter. The Rayon input is now highlighted. In the specification tree, select the "Rayon" parameter. Click **OK** and **Close**. The HeightCheck is added to the specification tree and, depending on the values assigned to the Height (Hauteur) and Radius (Rayon) parameters, a message can be displayed.

7. Double-click the HeightCheck relation twice in the specification tree. The relation below is displayed in the check editor:

   Hauteur >= Rayon.

The relations of a catalog must be instantiated one-by-one in a document.

Updating Relations Using Measures

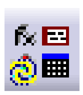
This task explains how to update relations using measures.

A relation using measures is to be updated when the  symbol is displayed opposite the relation in the specification tree.

Example

MeasureRule requires an update	
MeasureRule does not need to be updated	

To update the rules, proceed as follows:

- Click the  icon. To do this you must be in the Knowledge Advisor workbench.

  -or-

- Select the **Measure Update** command from the Relations node contextual menu. You can do this in any workbench.

All the document relations are then updated.

Controlling Relations Update

This topic provides the user with 2 short examples when working with relations in **Synchronous** and in **Update relation at global update command** modes.

- In the first example, the user updates a formula.
- In the second example, the user updates an equation.

- For a given relation, it is possible to determine that it only executes in Synchronous mode using the **Edit->Properties** command. Its evaluation can then be launched manually. The **Synchronous** mode enables the user to create synchronous relations, that is to say relations that will be immediately updated if one of their parameters is modified. Relations based on parameters are the only ones that can be synchronous.

- For a given relation, it is also possible to decide that its evaluation will be launched when the part is updated (**Update relation at global update command**). The relations can be asynchronous for 2 reasons:
  - The user wants the relations to be asynchronous
  - The relation contains geometry.

It is possible to change the update mode of a relation after it is created. To do so, proceed as follows:

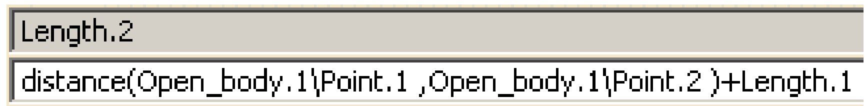
- Right-click the relation in the specification tree and select the **Properties** command.
- Check the **Synchronous** or the **Update relation at global update command** check box.

To know more about the **Synchronous** or the **Update relation at global update command** options, see the description of the Knowledge tab in the Infrastructure User's Guide.

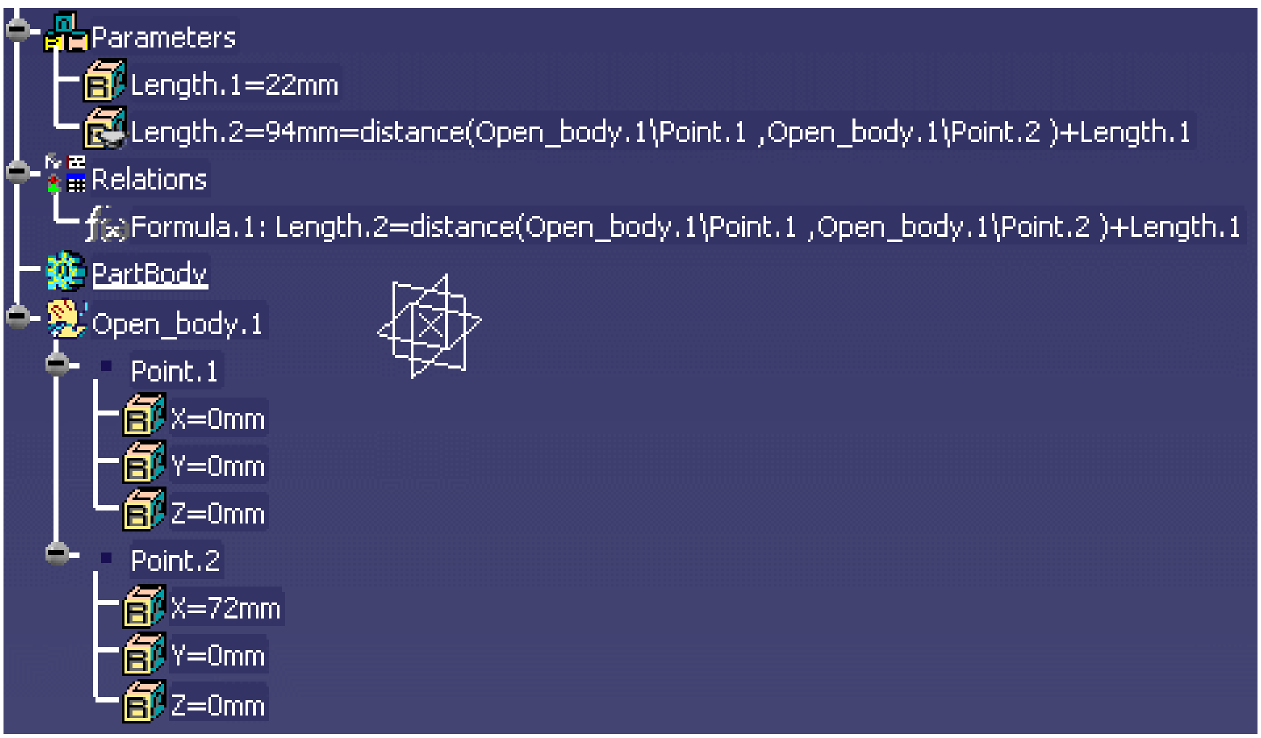
Updating a Formula

In this example, the document contains 2 parameters of **Length** type:

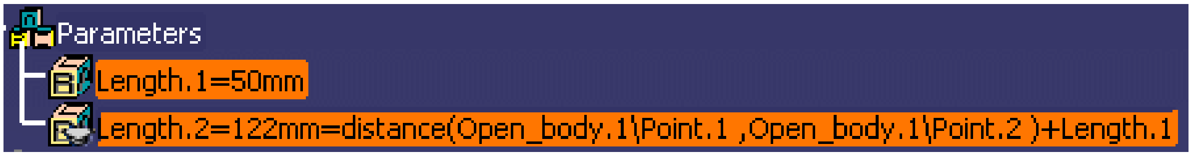
- Length.1 is valuated by the user
- Length.2 is valuated by a formula computing the distance between 2 points + Length.1.



1. Open the KwrUpdatingaFormula.CATPart file. The following image displays.



2. Expand the Relations node (if need be), right-click Formula.1 and select the **Properties** command.
3. In the **Properties** window, **uncheck the Update relation at global update command** check box. Click **OK** to validate.
4. Double-click Length.1. The Edit parameter window displays. Enter 50 in the value field. Click **OK** to validate. The value of Length.2 is updated automatically (See picture below).

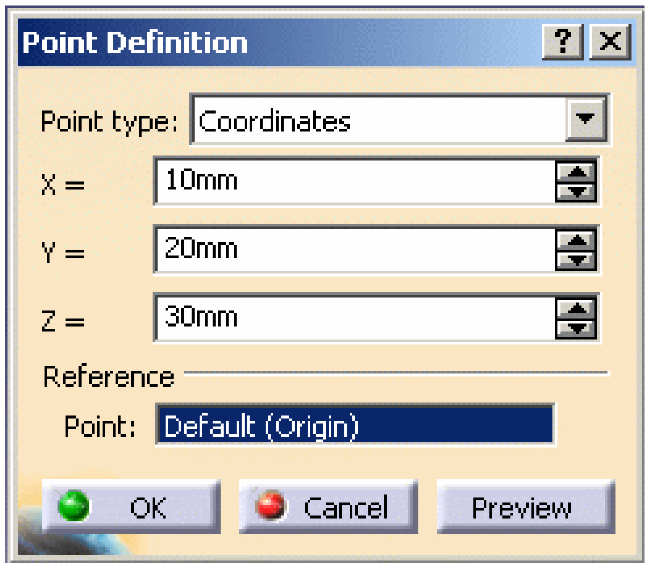


5. Right-click Formula.1 and select the **Properties** command.

6. In the Properties window, uncheck the **Synchronous** check box and check the **Update relation at global update command** check box. Click **OK** to validate.

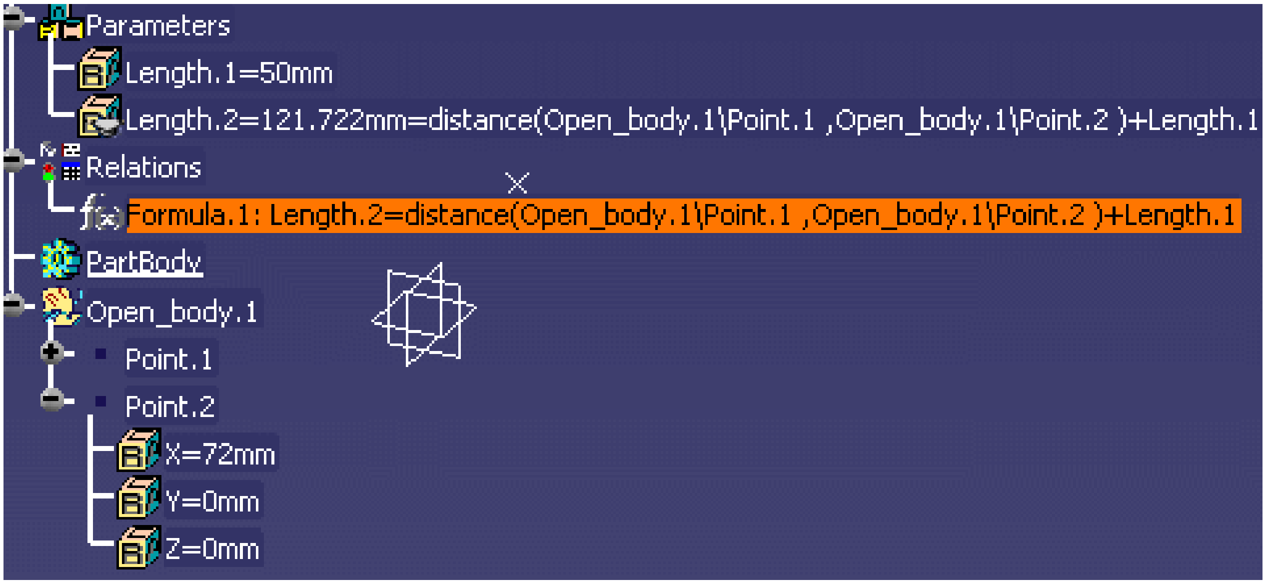
7. Double-click Point.2 in the specification tree or in the geometry.

8. Enter the following values in the **Point Definition** window and click **OK** when done:



The Formula is not updated and an update icon displays next to the formula.

9. Right-click Formula.1 and select the Local Update command. The formula and Length.2 are updated.

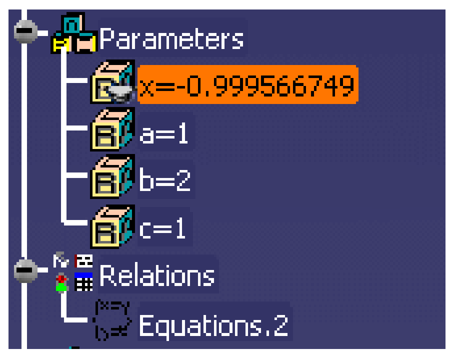


Updating an Equation

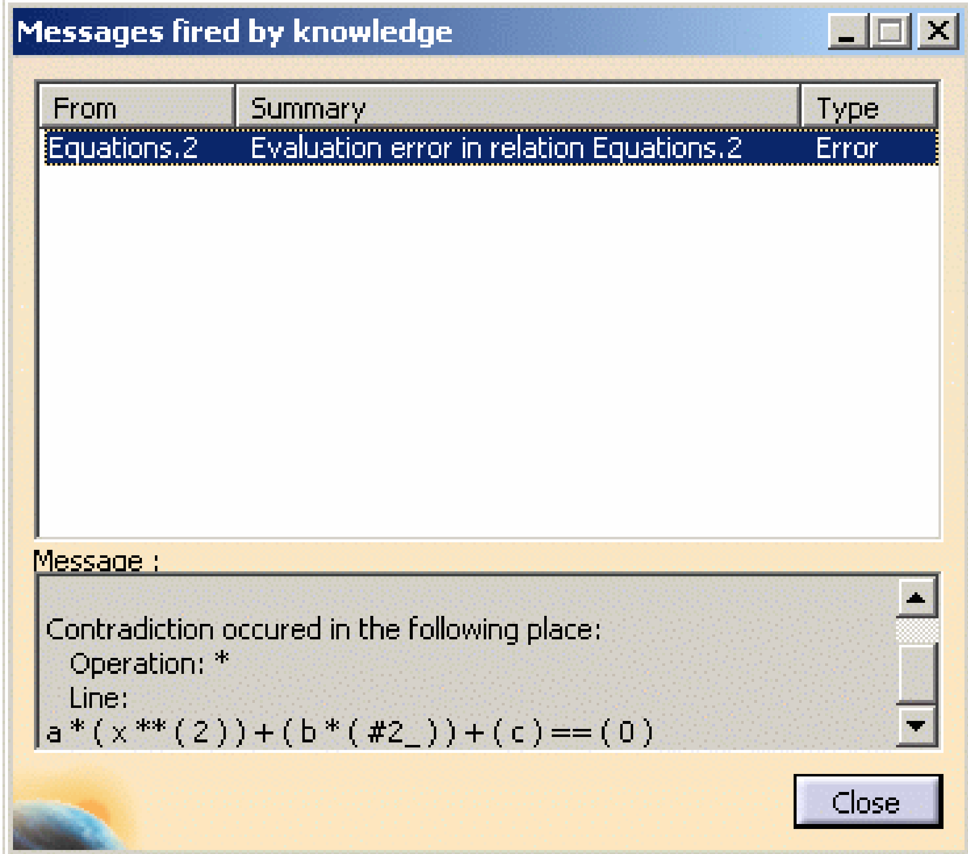
In this example, the user changes 3 parameters, a, b and c before solving the equation. If he uses the **Synchronous** mode, an error displays when modifying the parameters values since the update is launched at each modification. To ensure the stability of the equation, the update must be launched after the 3 parameters are changed. Thus the user needs to select the **Update relation at global update command** mode.

1. Open the KwrUpdatinganEquation.CATPart file. The following image displays:

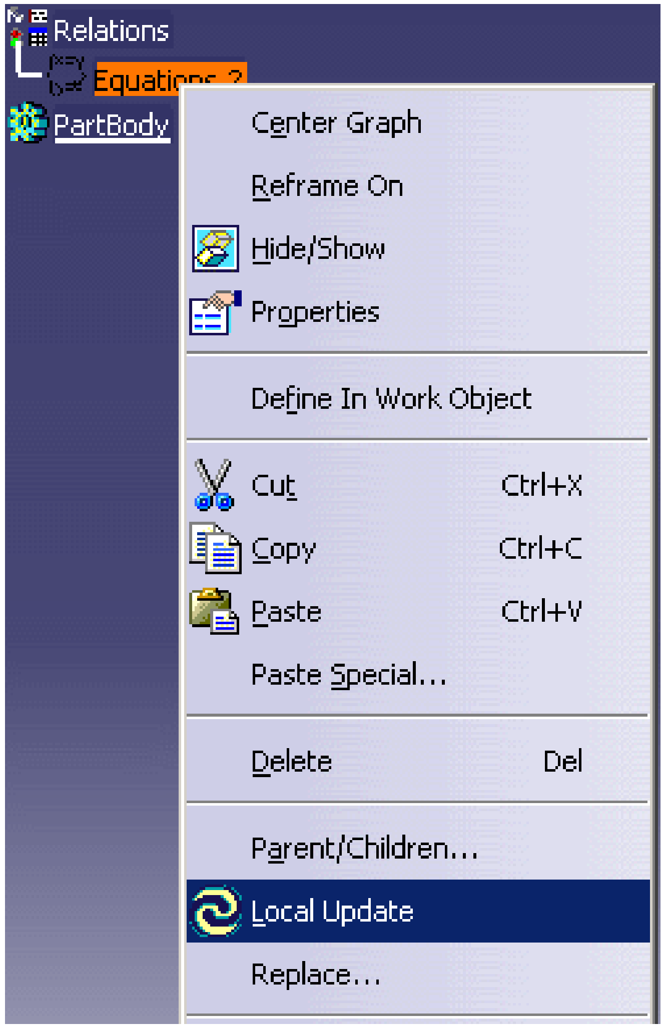
   This document contains 4 parameters. x is a parameter valuated by an equation based on the other 3 parameters (a, b, and c).



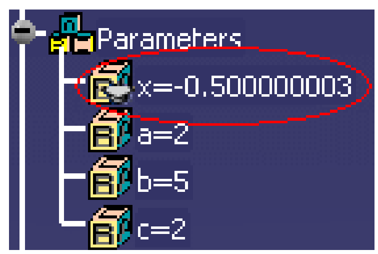
2. Expand the Relations node (if need be), right-click Equations.2 and select the **Properties** command.
3. In the Properties window, check the Synchronous check box and uncheck the Update relation at global update command check box. Click **OK** to validate.
4. Double-click a in the specification tree. The Edit Parameter window displays. Enter 2 in the value field. A message displays indicating that the equation cannot be solved and click **OK**.



5. Click **Close**.
6. Right-click Equations.2 and select the **Properties** command.
7. In the **Properties** window, uncheck the **Synchronous** check box and check the **Update relation at global update command** check box. Click **OK** to validate.
8. Double-click a in the specification tree. The Edit parameter window displays. Enter 2 in the value field. Click **OK**.
9. Double-click b in the specification tree. The Edit parameter window displays. Enter 5 in the value field. Click **OK**.
10. Double-click c in the specification tree. The Edit parameter window displays. Enter 2 in the value field. Click **OK**.
11. Right-click the equation in the specification tree and select the **Local udpate** command.



The equation is updated as well as the x value. For the equation to be solved, the **Update relation at global update command** option must be checked and the relation must be either integrated to the global update or updated manually using the Local Update command.



Using the Action Feature

This task explains how to use an action.

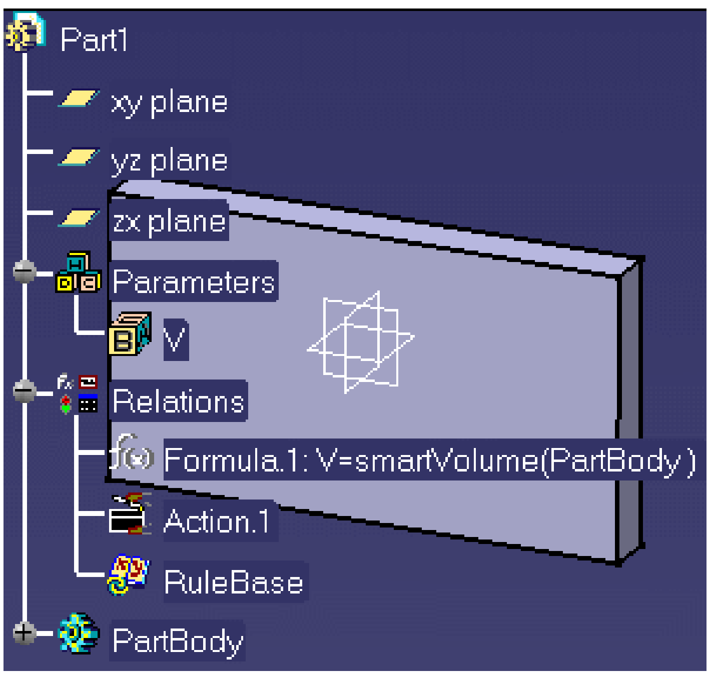
The scenario described below is made up of 3 major steps:

- You first create a pad containing an action.
- You store this action in a catalog
- You then import the action stored in the catalog into another CATPart product.

It is highly recommended to be familiar with the Part Design workbench to carry out this scenario.

1. Access the Part Design workbench and create a Pad or open the KwrAction.CATPart file.
2. Create a parameter of volume type and assign it a formula. To do so, proceed as follows:

Click the icon, select **Volume** in the scrolling list, click the **New Parameter of type** button and rename the Parameter (V in this scenario).



- Click the **Add formula button**. The formula editor opens.
- Under **Dictionary**, select **Part Measures**, and double-click **smartVolume**. Position the cursor between the parentheses and select PartBody. Click **OK**, **Yes** (when prompted for an automatic update) and **OK**.

3. Access the Knowledge Advisor workbench and click the Action icon ( ) to create an action. The Action editor opens. Enter the following script and click **OK**:

Inputs field	B: Body
Editor	B.Query("Pad","").Compute("+","Solid","smartVolume(x)",V) Message("Total volume of the pads under this body : #",V)

- The action created above searches for the pads contained in the selected body and computes the volume of these pads.
- To know more about Query and Compute, click here.
- To see the created .CATPart file, click here.

4. Save your file and store the created action in a catalog. To do so, proceed as follows:

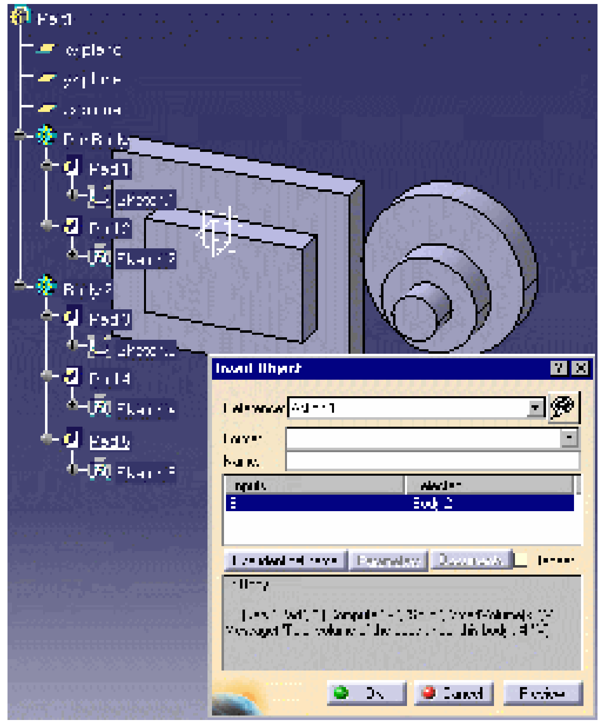
- From the **Start** menu, select **Infrastructure->Catalog Editor**. The catalog editor opens.
- Click the **Add Family** icon ( ), or select the **Insert -> Add Family...** commands from the main menu to display the Component Family Definition dialog box. Indicate the name of the family (ComponentFamily.2 in this scenario), and click **OK**.
- Double-click the ComponentFamily.2 family in the catalog structure and click the **Add component** icon ( ), or select the **Insert -> Add Component...** command to display the Description Definition dialog box.
- Click the **Select external feature** button. Go back to the geometry, select the Action.1 feature in the specification tree and click **OK**. Save your catalog. The action contained in your .CATPart file is now stored in the catalog you have just created.

5. Open the KwrReceiveAction.CATPart file.

6. Click the Catalog icon ( ) to import the action stored in the catalog. Click the **Browse another catalog** icon ( ), select your catalog, and click **Open**. Double-click ComponentFamily.2, and double-click Action.1.

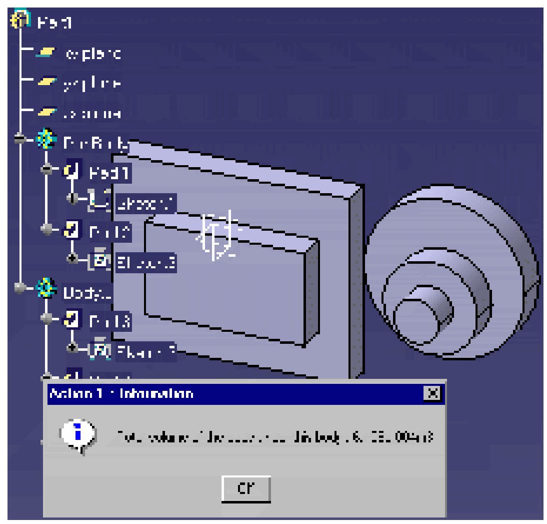
The **Insert object** dialog box opens.

(Click the graphic opposite to enlarge it.)



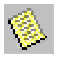
7. Select PartBody: the imported action displays the volume of the pads contained in this body.

(Click the graphic opposite to enlarge it.)



8. Select Body.2: the imported action displays the volume of the pads contained in this body.

Working with the List Feature



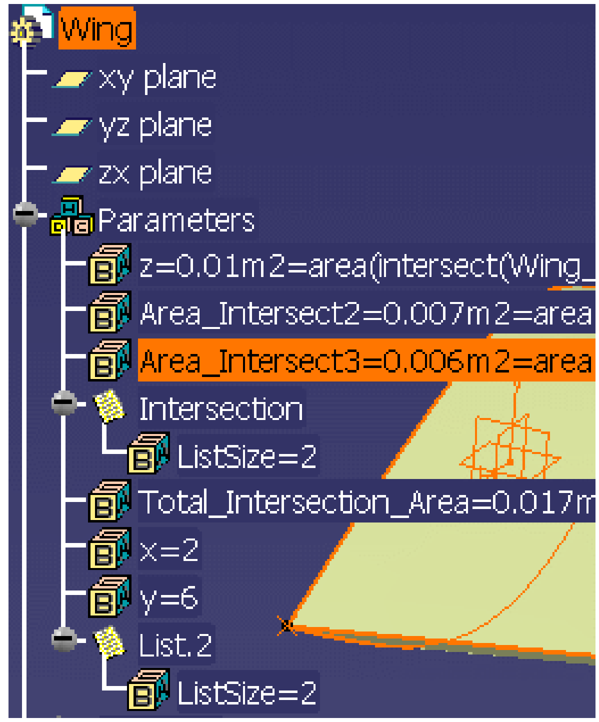
Select the List icon to create a list of features that will be located under the Parameters node in the specification tree. Clicking this icon opens the List edition window.

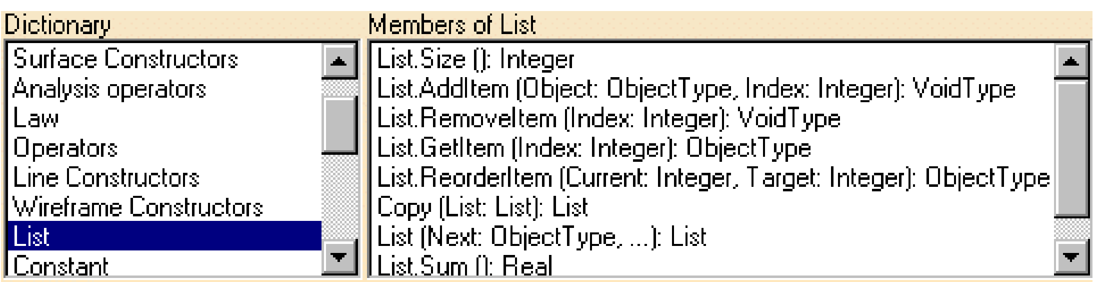
List features can be used to manage lists of objects or parameters. These lists can be edited interactively. The List edition window enables the user to sort items automatically and to specify the type of objects authorized.

The list feature is integrated in the update mechanism, the size of the list is computed automatically (it is provided with functions designed to compute sums, areas, costs...).

The list feature can be manipulated through the language to:

- Create list
- Copy the content of a list into another one
- Add and remove elements
- Get an element
- Retrieve values from the list
- Move elements of the list to another position

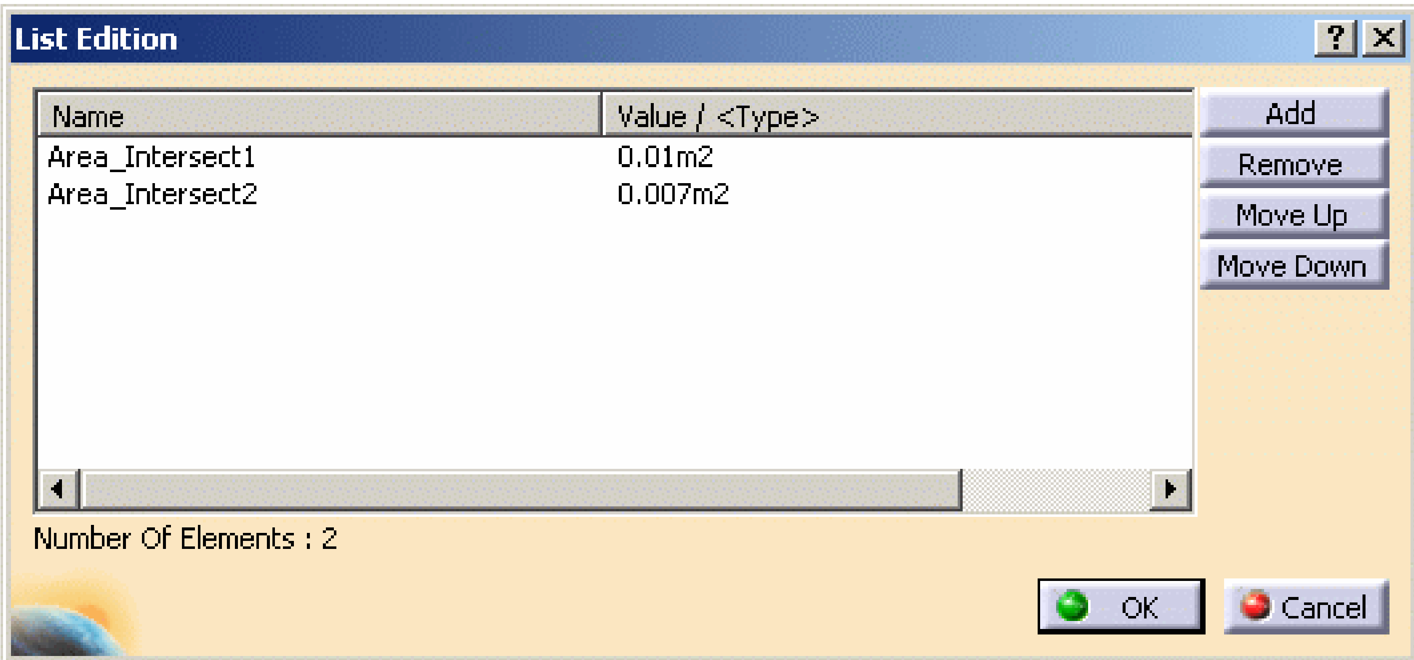




Creating a List
Using the List Edition Window

Using the List Edition Window

The List Edition window enables the user to manage the objects he wants to add to the list he is creating. It can be accessed by clicking the List icon ( ).



The window contains four different buttons and is made up of 2 columns.

Name	Column indicating the name of the list.
Value/<Type>	Column indicating the value of the list or the associated type.
Add	Enables the user to add the items he selected in the specification tree or in the geometry to the list.
Remove	Enables the user to remove items from the list.
Move Up	Enables the user to move up items in the list.
Move Down	Enables the user to move down items in the list.

The **Number of Elements** field displays the number of items contained in the list.

There are 4 different types of lists:

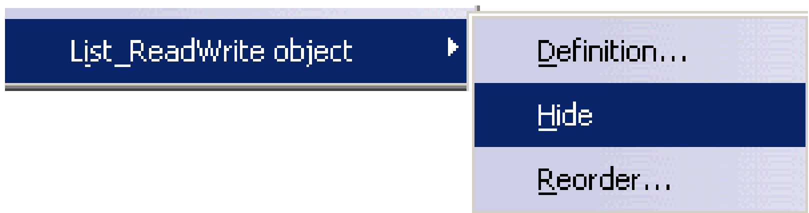
- Not seen lists: These lists are created by CAA users and cannot therefore be modified interactively by the user (they are not displayed). In this case, the buttons of the List dialog box do not display.

- Read only lists: These lists are created by CAA users and cannot therefore be modified interactively by the user (they are in read only mode).

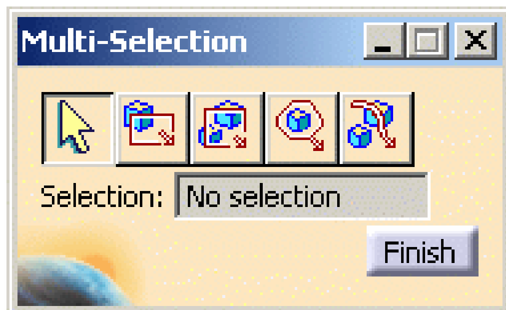
- Read/Write lists: These lists are created by CAA users, can be edited by the end user but they cannot be deleted.

- User lists: These lists can be edited and deleted by the end user.

- Like for any other parameter, it is now possible to hide and reorder lists. To do so, right-click the list and select the **Hide** or **Reorder...** commands.



- When clicking the List icon to create a list, the Multi-Selection panel now displays. To know more about this panel, see the *Infrastructure User's Guide*.



If you select an item in the List, and click another item in the specification tree or in the geometry, and click **Add**, the List item will be replaced with the one you have just added.

Creating a List

This task explains how to use the List Feature.
In the scenario described below, the user will manipulate a plane wing to which he will add planes and intersections. He will then create parameters and formulas to calculate the surface of the intersections and will create a list that will compute the total area of the intersect sections.

To know more about the List Edition window, see Using the List Edition Window.
To know more about the List Feature, see Working with the List Feature.

1. Open the KwrPlaneWing.CATPart file.

2. Access the Generative Shape Design workbench.

3. Create three planes. To do so, proceed as follows:

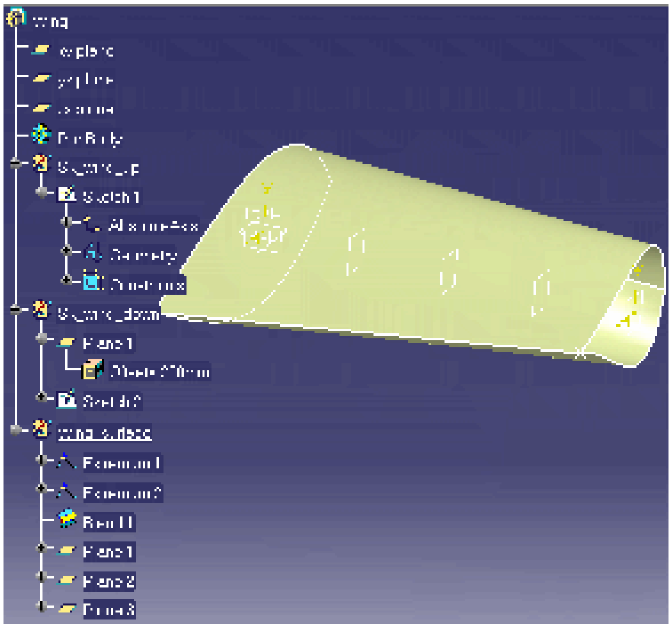
- Click the Plane icon in the tool bar.
- a) The **Plane Definition** dialog box opens.

  b) In the **Plane type** area, enter the yz plane (select it in the geometry or in the specification tree): The yz plane is displayed in the **Reference** area.

  c) Indicate the required offset in the **Offset** field (-50mm for example). Click **OK**.

  d) Repeat this operation twice with offsets of -100 and -150mm.

  (Click the graphic to enlarge it)



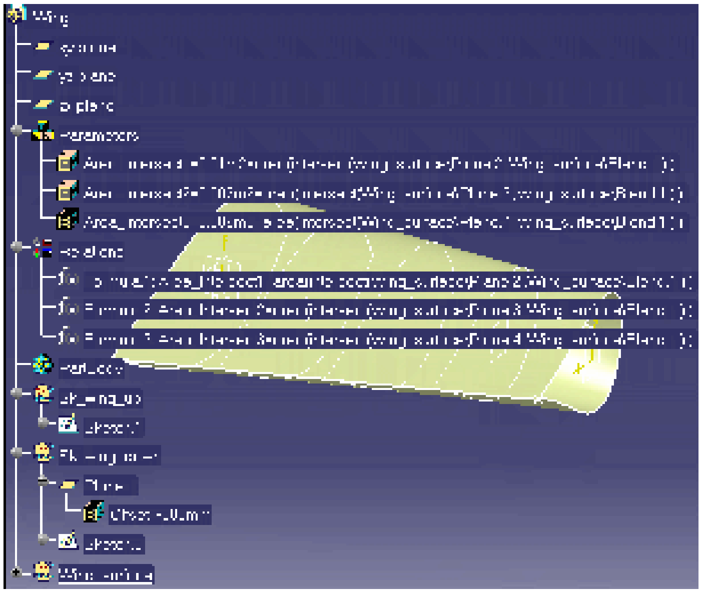
4. Add formulas to calculate the intersection surface of the planes with the blend. To do so, proceed as follows:

- Click the  icon. The Formulas Editor opens.
- Select **Area** in the scrolling list and click the **New parameter of type** button. Change the name of the parameter to Area_Intersect1 and click the **Add Formula** button. The Formula editor opens.
- In the **Dictionary**, select **Measures**, double-click **area(Surface, ...):Area**.

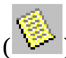
- Position the cursor between the parentheses, select **Wireframe constructors** in the Dictionary, and double-click **intersect(Surface,Surface):Curve**.

- Position the cursor before the coma and select Plane.2 in the specification tree (or in the geometry) then select Blend.1 in the specification tree. Click **OK**.

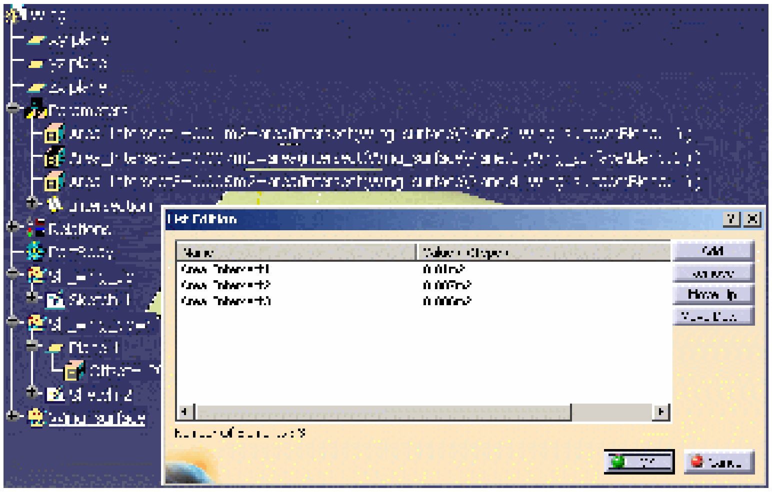
- Repeat the above steps for Area_Intersect2 and Area_Intersect3 by selecting Plane.3 and Plane.4.



(Click the graphic to enlarge it)

5. Access the Knowledge Advisor workbench, and click the List icon (). The List Edition window opens.

6. Select the 3 parameters located under the Parameters node in the specification tree, and click the **Add** button. Click **OK**. Rename List.1 to Intersections for example.



(Click the graphic to enlarge it)

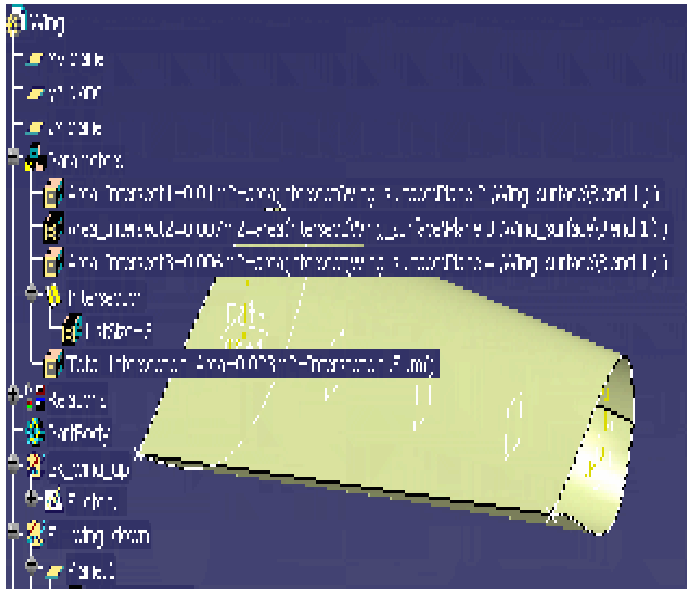
7. Click **OK**. The list is added to the parameters.

8. Add a formula that will compute the area of the 3 planes intersections with the blend. To do so, proceed as follows:

   - Click the  icon. Select **Area** in the scrolling list, click the **New parameter of type** button, rename the parameter to Total_Intersection_Areas and click the **Add formula** button. The Formula editor opens.

- Click the **List** (Intersections in this scenario) in the specification tree: the name of the list is displayed in the editor.
  Under Dictionary, select **List**, and double-click **List.Sum(): Real** in the **Members of List** area.
  Click **OK** twice.

  The area of the 3 planes intersections with the blend is automatically calculated.



(Click the graphic to enlarge it)

9. Edit the list content and re-compute the total area. To do so, proceed as follows:

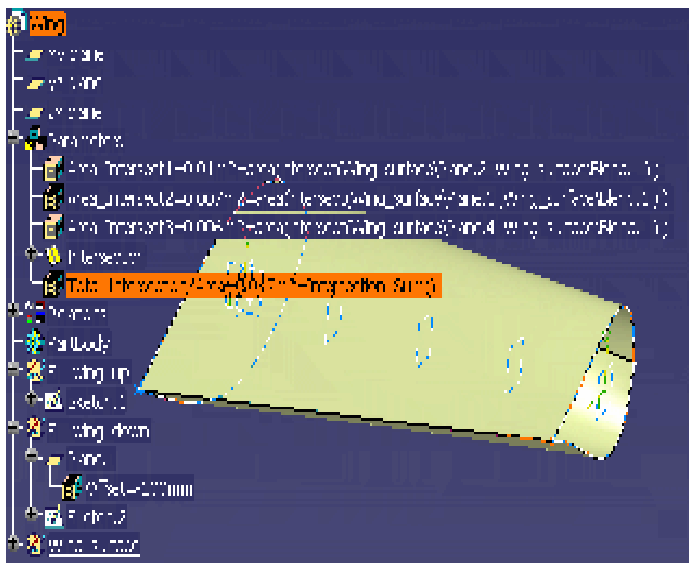
- Double-click the list (Intersections) in the specification tree: the **List** edition window opens. Select Area_Intersect3, click the **Remove** button, and click **OK**.

- Right-click the Total_Intersection_Areas parameter and select **Local Update**.

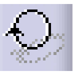
  The area of the remaining 2 planes intersections with the blend is automatically calculated.

  (Click the graphic to enlarge it)

  Click here to display the result of this scenario.



Working with the Loop Feature



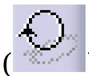
Select the Loop icon to create a loop.

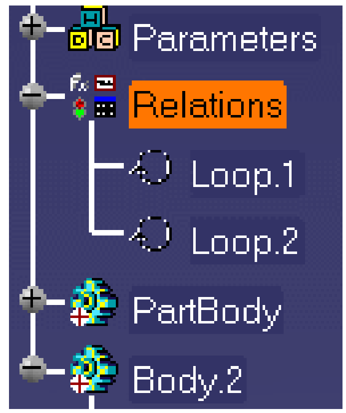
Introducing the Loop Feature

Note that a KWA license is required to execute loops.

Loops use the Generative Knowledge language to drive the creation, modification and deletion of a set of features. This functionality enables the user to:

- Select inputs in the definition of the loop
- Define several contexts in the loop action
- Include the loop into a powercopy

It can be accessed by clicking the **Loop** icon ().

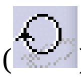


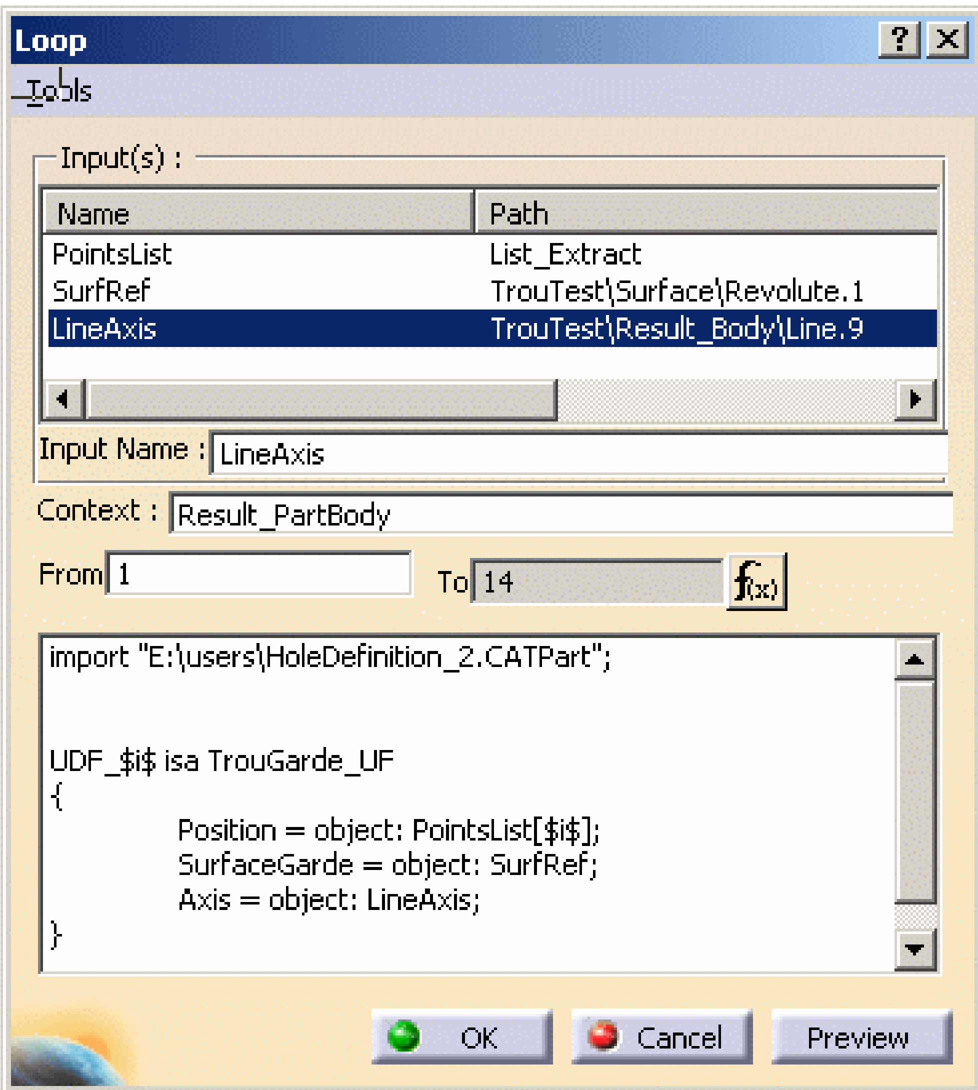
- A loop is stored in the resulting model as a feature on its own. A change in its specification will drive the expected modification in the model.
- A loop can be instantiated through a PowerCopy implying a significant simplification of use and re-use.

Getting Familiar with the Loop Edition Window and Menus

- The Loop Edition Window
- The Loop Tools Menu
- The Loop Contextual Menu

The Loop Edition Window

The Loop Edition window is displayed when you click the Loop icon () in the **Control Features** tool bar.



Input(s)

This field enables the user to select the features that he wants to use in the specification tree or in the geometry. The selected features are those that will be used in the loop body.

To deselect items from the Inputs list, click them in the specification tree or in the geometry.

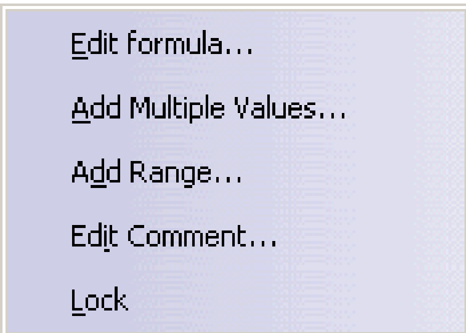
The **Input Name** field enables the user to rename the inputs that he selected. In this case, this name will be used in the loop body.

Context

This field enables the user to define the application context of the loop. It can be any V5 feature. To select the context, click the Context field once, then click the item in the specification tree.

Iterators

The **From ... To** fields enable the user to define the number of times that the loop will operate. When defining the ranges, the user can right-click the **From...** and the **To** fields to access the contextual menu.



- The **Edit formula...** command enables the user to access the Formulas editor and to create a formula that will apply to the loop operation. To know more, see Creating a Formula.
- The **Add Multiple Values...** command enables the user to add multiples value. To know more, see Switching between Simple and Multiple Values After Creating a Parameter.
- The **Add Range...** command enables the user to add a range.
- The **Edit Comment...** command enables the user to add a comment.
- The **Lock...** command enables the user to lock this parameter. To know more, see Locking and Unlocking a Parameter .

Note that:

- The step is one in the **From... To** fields.
- Both bounds are included when the loop runs.

Editor

The Editor enables the user to enter the loop syntax. The language to use in this editor is the scripting language. To know more about the syntax to be used, see Using the Scripting Language.

The Tools menu

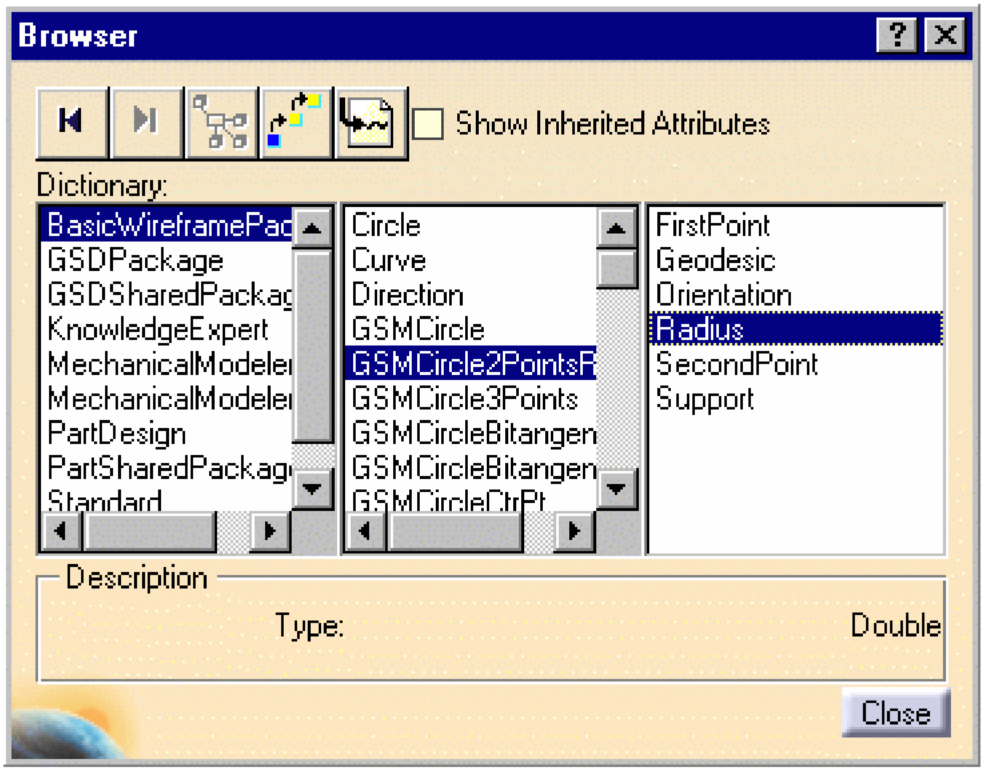
The **Tools->Object Browser ...** command enables the user to access the Object Browser. This browser contains the types and attributes that are part of the scripting syntax.

The object browser guides you when writing a script. It allows you to access the keywords, operators and feature attributes that can be used when working with the loop features.

The packages displayed in the left part of the browser are those you selected from the **Tools->Options...** command.

To add or remove packages, proceed as follows:

1. Select the **Tools->Options...** command to open the Options window, then select **General->Parameters and Measure**, and click the Language tab.
2. In the **Language** field of the **Knowledge** tab, check **Load extended language libraries** and select the libraries.



From this window, you can manipulate the list of objects supported by the script using their attributes...

- The left part of the browser displays the available packages.
- The central part displays the list of objects belonging to this category.
- The right part displays the attributes allowing you to manipulate these objects (if any).

... and write loop bodies (see example below):

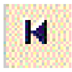
```
Circle0 isa GSMCircle
      {
         CircleType = 0;
         TypeObject isa
GSMCircleCtrRad
         {
         Center = object:
..\..\..\Construction_Body\Point.2;
         Support = object:
..\..\..\Construction_Body\Extrude.1;
        Radius = 180mm;
         }
         StartAngle = -100deg;
         EndAngle = 100deg;
        }
```


The ***Back*** icon.
To return to your last interaction in the wizard. Has no action on the script editor.


The ***Forward*** icon.
To go forward to your next interaction in the wizard when moving through a series of interactions.


The ***Attribute Type*** icon.
This icon is not available in the current version of the product.

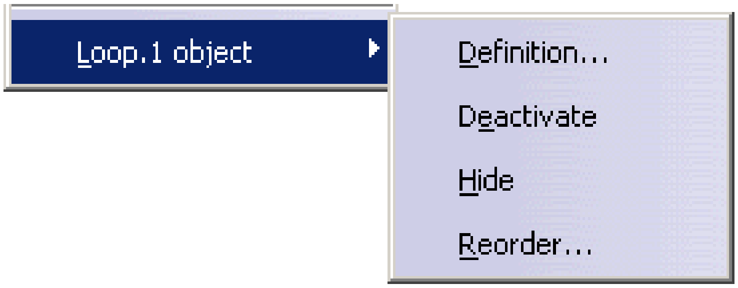

The ***Inheritance*** icon.
To return to the root object.



The ***Insert*** icon.
To insert the object name in the script.

The Loop Contextual menu

You can access the Loop contextual menu by right-clicking the loop in the specification tree.

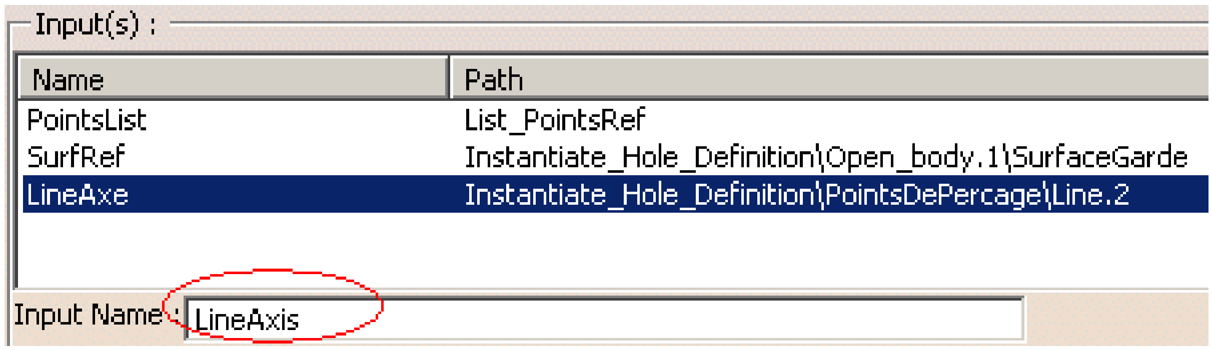


- The **Definition...** command enables you to access the Loop Edition window.
- The **Deactivate...** command enables you to deactivate the loop. In this case an icon indicates that the loop is disabled. To enable it, right-click it and select the **Loop activate...** command.
- The **Hide** command enables you to hide the loop. In this case, it will not display in the specification tree.
- The **Reorder...** command enables you to reorder the loops.

Declaring Input Data

The Input Data are the data that will be used in the body of the loop and potentially be changed when instantiating the loop. To select them, click them in the specification tree or in the geometry.

The Input Name field enables you to change the name of the input.



Note that the input can be a list made up of strings. It allows to instantiate features whose types are defined by the list items. To get an example see the following files:

- KwrPartUDFs.CATPart: This file contains the 4 UDFs definitions.
- KwrInstantiateUDFs.CATPart: This file contains the loop that instantiates the features. The inputs of the loop are made up of 2 lists. The first list contains parameters of String type (the names of the UDFs to be instantiated). The second list contains parameters of Length type (to valuate the UDFs published parameters: PosX).

Defining the Context

To create a loop, the user needs to define the context, that is to say the object (PartBody, Geometrical set, Pad, Relations, Parameters node or any feature) that will contain the items created by the loop. There are 3 different ways to define the context.

Using the Context field

To define the context of the loop, you may use the Context field of the Loop edition window.
To do so, click the Context field, and select an object in the specification tree. In the picture opposite, the user selected a pad in the specification tree.

Using an existing Document

It is possible to use an existing .CATPart or .CATProduct document.

Using the context keyword

When creating elements that need to be located in different bodies, the user can change the context he defined in the Context field and use the **Context** keyword to define new contexts in the loop body.

```
Pa1_$i$ = 12mm;
Pt1_$i$ isa GSMPoint{PointType = 0;TypeObject isa GSMPointCoord{ X  = 20mm;}}

context ListeContext[2]
Pa2_$i$ = 1mm;
Pt2_$i$ isa GSMPoint{PointType = 0;TypeObject isa GSMPointCoord{ Y= 15mm;}}


context ListeContext[3]
Pa3_$i$ = 3mm;
Pt3_$i$ isa GSMPoint{PointType = 0;TypeObject isa GSMPointCoord{ Z= InputHauteur;}}
```

The action script should not start with the context keyword since the first context is defined in the Context field.

Sample: KwrLoopMulticontext.CATPart (to launch the loop, activate the loop.)

Using the Scripting Language

Introducing the Scripting Language

The Scripting Language is a declarative way of generating V5 Features. It allows users to:

- Describe objects using a very simple script language.
- Use 3D geometric features (sketches, parts, ...).
- Use parameters on features including *formulas*.
- Use related positioning & orientation constraints.
- Generate the corresponding V5 models (features, documents, User Features,...)
- Enter the body of their loops in the Loop Edition window in Knowledge Advisor.

Action Script Structure

An action script is written in text format and is organized in blocks consisting of related sets of statements. A block consists in an instruction designed to create an object followed by a set of statements surrounded by braces (**{ }**). Statement blocks can be nested and the most enclosing one within a script corresponds to the document creation.

A document is made up of a hierarchy containing objects, their properties and the features they own. An action script reflects this object hierarchy.

Example

```
UDF_$i$ isa Hole_UDF  1
{
            Position = object: PointsList[$i$];
        2   SurfaceGarde = object: SurfRef;
            Axe = object: LineAxe;
            Garde = 3mm;
}
```

In the script opposite, the inputs and the published parameters (**2**) of the instantiated UDF "Hole_UDF" (**1**) are nested between braces {}.

Object Properties

- An object is created by default with some property values. These properties are defined or re-defined within the braces just following the object declaration (isa keyword).
- Unless otherwise specified, the units are IS units.
- When defining properties, the semicolon **;** is a terminator (see example below). The properties might be object attributes (1), attributes needed to define a type displayed in the Object browser (2) or aggregated objects (3).

```
UDF_$i$ isa Hole_UDF
{
            Position = object: PointsList[$i$];
            SurfaceGarde = object: SurfRef;
            Axe = object: LineAxe;
            Garde = 3mm;
}
```

1

In the script above, the properties are the inputs of an instantiated UDF.

```
Line.1 isa GSMLine
{
    LineType = 0;
    TypeObject isa GSMLinePtPt
    {
      FirstPoint = object: Point.1;
      SecondPoint = object: Point.2;
      Length1 = 500mm; \\ optional
      Length2 = 435mm; \\ optional
    }
  }
```

2

In the script above, the properties are the attributes required to create a point to point line.

```
MyBody isa OpenBodyFeature
{
  Box isa Pad
       {
        }
}
```

3

In the script above, the Pad object is aggregated below the OpenBodyFeature object.

Keywords

isa Keyword

Definition

Enables the user to create a typed object or instantiate an object.

Syntax

- **ObjectName isa ObjectType**

or

- **ObjectName isa InstanceName**

where:

- ObjectName is the name of the object to be created.
- ObjectType is the type of the object to be created.
- InstanceName is the name of the object to be instantiated.

Example

```
import "E:\GPS.CATPart";
 myGps isa CATPart
      {
      myPart isa Part
             {
                PartB isa BodyFeature
                      {
                        S0 isa Sketch.0 {} //Instance name
                        pad0 isa Pad("S0") //Object type
                       }
              }
      }
```

context Keyword

Definition

Enables the user to define in which part of the specification tree the object will be created. The context keyword may be of use in 2 different cases:

- It can indicate a document to be used. In this case, the "..." are used.

```
context "Mypart.CATPart"
MyPart isa Part { }
```

- It can reference an object contained in the document. In this case the path needs to be specified (between `...`).

```
context `My.CATPart\MyPart\PartBody`
CC isa Cylinder { }
```

Syntax

- **context "Mypart.CATPart"**

or

- **context `My.CATPart\MyPart\PartBody`**

from Keyword

Definition

Allows the user to copy a document from an existing document without maintaining any link.

Syntax

DocumentName isa *DocumentType* from *FilePath*

where:

DocumentType is either CATProduct, CATPart or model.
FilePath is the full path of the initial document.

To enter a file path you can:

Use the Insert File Path command from the contextual menu

Example

See Defining the Context

import Keyword

Definition

Specifies a document file (.CATPart or .CATProduct) containing definitions to be reused or redefined in the document to be generated. All the features and feature values in the imported file become available to the document to be generated.

Importing a document is:

- Of interest whenever you want to retrieve a consistent set of definitions from an already existing document (for a UDF definition for example.)
- Required whenever you need to create a feature from a sketch (the script language does not allow you to specify a sketch).

Syntax

There are 2 ways to specify the file path:

- **import "File path";** : Indicate the path of the file to be imported: `import "E:\users\kwecx\Models\PartImport.CATPart" ;`. Note that you should enclose the path within quotation marks and end the import statement with a semicolon (;).
- **import "File Name";** : Indicate only the name of the file to be imported if this file is located in the same directory as the document containing the loop: `import "PartImport.CATPart";`. Note that:
    - You should enclose the document name within quotation marks and end the import statement with a semicolon (;).

- The file to be imported should be located in the same directory as the document containing the loop.
- The document containing the loop should be saved.

To specify a file to be imported, you can use the 'Insert File Path' command from the contextual menu. Selecting this command displays a file selection panel. Quotation marks are automatically included but not the semicolon.

Example

```
import "E:\users\mei\R12\PES\kwr\loop\Hole_Definition.CATPart";

UDF_$i$ isa Hole_UDF
{
        Position = object: PointsList[$i$];
        SurfaceGarde = object: SurfRef;
        Axe = object: LineAxe;
        Garde = 3mm;
}
```

publish Keyword

Definition

Enables the user to assign an object a name that will be used in the script.

Syntax

publish "!xxx" **as** yyy ;

Where:

- xxx is the name of the object to be published. To select this object, it is highly recommended to use the contextual menu.
- yyy is the name you want to assign to this object

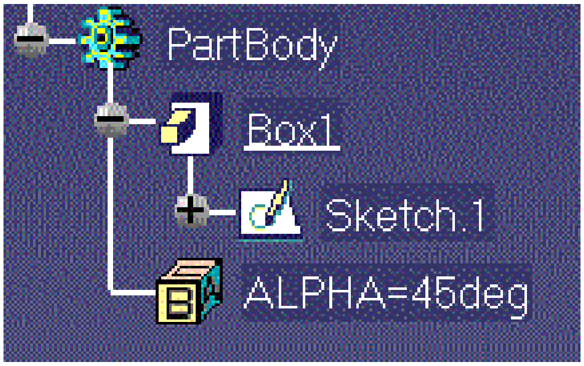
Example

```
Publi isa CATProduct
{
    Publi isa Product
    {
         P isa Product
         {
             P1 isa Part
             {
                 PartBody isa Feature
                 {
                     Pa isa Pad{}
                 }
             }
         publish "Publi/P/!Selection_RSur:(Face:(Brp:(Pa;2);None:());Pa)" as mypadface; /*publishes
         the face of a pad under the name "mypadface"*/
         }
         Q isa Product
         {
           Q1 isa Part
           {
               PartBody isa Feature
               {
                   Cy isa Cylinder{}
               }
           }
          publish "Publi/Q/!Selection_RSur:(Face:(Brp:(Cy;2);None:());Cy)" as mycylinderface;
          //publishes the face of a cylinder under the name "mycylinderface"
          }
          assembly constraints: contact("P\toto","Q\tutu");
      }
}
```

Variables

Variables are declared explicitly in your script. These variables are displayed as parameters in the specification tree.

```
ALPHA = 45 deg;
```



Unlike in most script languages, a variable scope is not really determined by where you declare it. From anywhere in your action script, you can access a variable by using the ..\.. and ? operators. After the script is finished running, the variable declared in your script still exists as a document parameter.

Operators

Arithmetic operators

+ Addition operator (also concatenates strings)

- Subtraction operator

* Multiplication operator

/ Division operator

( ) Parentheses (used to group operands in expressions)

= Assignment operator

? (Question Mark in Formulas)

Definition

In a formula, specifies that the parameter value to be applied is the first parameter value found when scanning the specification tree from the formula to the top of the specification tree.

Sample

KwrLoopRelativePath.CATPart

(Relative Path in Formulas)

Definition

Defines where the value of a parameter used as an argument in a formula is to be read. A single.. exits the statement block where the formula is defined. The parameter value applied in the formula is then the one defined in the parent feature scope.

Sample

KwrLoopRelativePath.CATPart

Using The Get... Commands

The commands described in this section are the ones the user can access when using the Loop Editor and right-clicking in the Editor window.

When creating a loop containing the path of a feature contained in the specification tree, it is highly recommended to use the **Get Feature** command to retrieve the internal name of this feature.

- Using the Get Axis Command
- Using the Get Edge Command
- Using the Get Surface Command
- Using the Get Feature Command
- Using the Insert File Path Command

The 'Get Axis' Command

This task explains how to create a chamfer by using the **Get Axis** command. This command enables the user to interactively capture the generic name of an axis and to insert it into the script instead of keying it in.

1. Click the **Loop** icon ( ) and enter 1 in the **To** field.
2. In the Script Editor, enter the following script and click **OK**. A pad is created.

```
myChamferDocument isa CATPart
{
        myPart isa Part
        {
                PartBody isa BodyFeature
                {
                        P isa Pad
                        {
                        }

                }
        }
}
```

3. From the **Window** menu, select **Cascade**.
4. Under the P isa Pad block, add F isa Chamfer(){}, position the cursor between the parenthesis, then right-click to open the contextual menu and select the **Get Axis** command, and select an edge in your geometrical surface. The script should be as

follows:

```
myChamferDocument isa CATPart
{
       myPart isa Part
       {
                PartBody isa BodyFeature
                {
                     P isa Pad
                     {
                     }
                     F isa Chamfer("Edge:(Face:(Brp:(P;0:(Brp:(Sketch.1;2)));
                     None:());Face:(Brp:(P;0:(Brp:(Sketch.1;3)));None:());
                     None:(Limits1:();Limits2:()))"){}
                }
        }


}
```

5. Click the **OK** button. The chamfer is created.

The "Get Edge" Command

This task explains how to create a chamfer by using the Get Edge command. This command enables the user to interactively capture the generic name of an edge and to insert it into the script instead of keying it in.

1. Click the **Loop** icon ( ) and enter 1 in the **To** field.
2. In the Script Editor, enter the following script and click **OK**. A pad is created.

```
myChamferDocument isa CATPart
{
       myPart isa Part
       {
                PartBody isa BodyFeature
                {
                       P isa Pad
                       {
                       }

                }
       }
}
```

3. Under the P isa Pad block, add F isa Chamfer(){}, position the cursor between the parenthesis, then right-click to open the contextual menu and select the **Get Edge** command, and select an edge in your geometrical surface. The script should be as

follows:

```
myChamferDocument isa CATPart
{
        myPart isa Part
        {
                PartBody isa BodyFeature
                {
                   P isa Pad
                   { }
                   F isa Chamfer("Edge:(Face:(Brp:(P;0:(Brp:(Sketch.1;2)));

None:());Face:(Brp:(P;2);None:());None:(Limits1:();Limits2:()))"){}
                }
        }
}
```

4. Click the **Generate** button. The chamfer is created.

The "Get Surface" Command

This task explains how to create a sketch on an existing face by using The **Get Surface** command. This command enables the user to interactively capture the generic name of a surface and to insert it into the script instead of keying it in.

1. Open the KwrGetSurface.CATPart file.
2. Access the Knowledge Advisor workbench, and click the Loop icon. Enter 1 in the **To** field
3. Enter the following script:

```
import "f:\cube.CATPart";
myFaceDocument isa CATPart
{
        myPart isa Part
        {
              PartBody isa BodyFeature
              {
              P isa Pad{}
              S isa Sketch.1()
```

4. Position the cursor between the two parentheses of the last line of the above script, right-click to open the contextual menu and select the **Get Surface** command.
5. Select the face whose name you want to capture. The full name is inserted at the cursor location. Enter the end of your script. In our example, the final script is as follows:

```
import "f:\PktGetSurface.CATPart";
myFaceDocument isa CATPart
{
        myPart isa Part
        {
            PartBody isa BodyFeature
            {
            P isa Pad{}
            S isa Sketch.1("Face:(Brp:(P;0:(Brp:(Sketch.1;2)));None:())")
            {
            }
            }
        }
}
```

The "Get Feature" Command

This task explains how to use the Get Feature command. This command enables the user to interactively capture the generic name of a surface and to insert it into the script instead of keying it in. In the task below, the user generates a line.

1. Open the KwrGetFeature.CATPart file.

2. Double-click the loop located below the Relations node and insert the following code into the editor:

```
Line_Pt_Pt isa GSMLine
{
LineType = 0;
TypeObject isa GSMLinePtPt
{
FirstPoint = object: // Using GetFeature to Select the FirstPoint
SecondPoint = object: // Using GetFeature to Select the SecondPoint
}
}
```

3. Position the cursor after FirstPoint = object: and select the **Get Feature** command in the contextual menu.

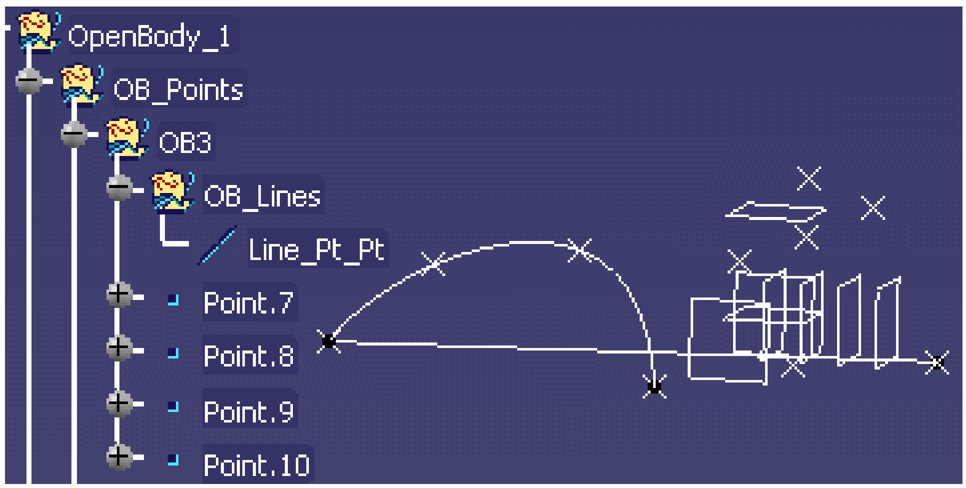
4. Click a point in the geometry and add a semi-colon (;) at the end of the line.

5. Position the cursor after SecondPoint = object: and select the Get Feature command in the contextual menu.

6. Click another point in the geometry and add a semi-colon (;) at the end of the line. Your script should now look like the one below:

```
Line_Pt_Pt isa GSMLine
{
LineType = 0;
TypeObject isa GSMLinePtPt
{
FirstPoint = object: Open_body.1\Open_body.2\Open_body.3\GSMPoint.10;
SecondPoint = object:Open_body.1\Open_body.2\GSMPoint.4;
}
}
```

7. Click **OK**. A new line is generated.



The 'Insert File Path' Command

This task explains how to use the Insert File Path command. This command is one of the methods you can use to specify a path in a script.

When writing a script, you have to specify a file path when you import a file, see the import keyword.

1. Access the Script Editor and enter any instruction requiring a file path specification (import in the example below).

2. Position the cursor where the path is to be specified.

3. Right-click and select the **Insert File Path** command from the contextual menu.

4. In the dialog box which is displayed, select the appropriate file. Click **Open** to go back to the script editor.

   The full path is inserted at the cursor place. Check that the statement is ended by a semi-colon.

Comments

Multi-line comments (/* ... */) are supported. A single-line comment begins with a pair of forward slashes(//).

Note that DBCS characters are not supported as comment.

Example

```
Sphere1 isa Sphere // Creates a sphere
      {
         // Valuates the Radius property
         Radius = 15.0 ;
      }
```

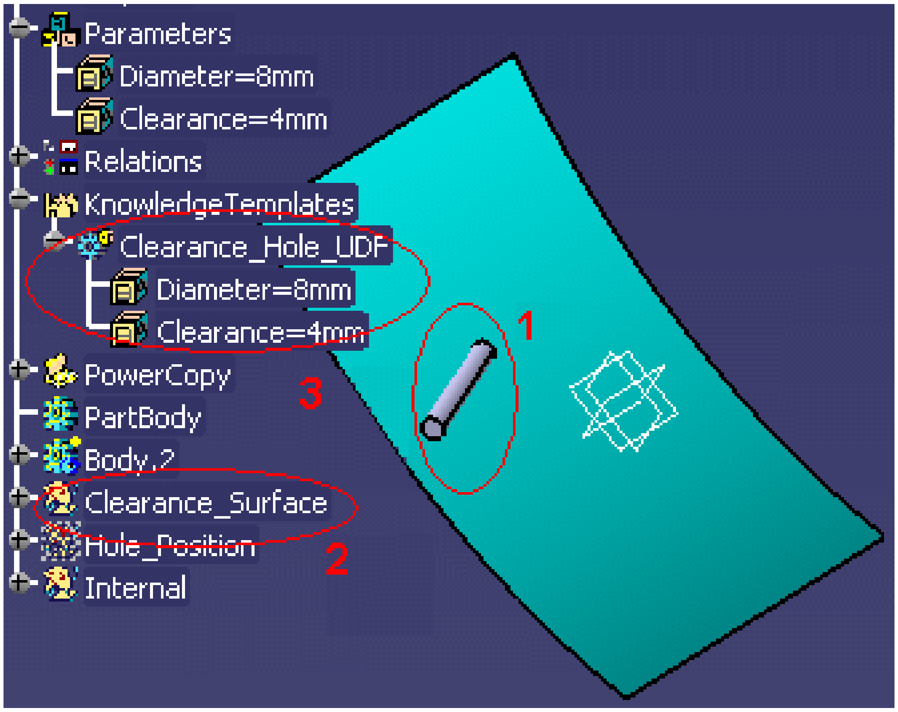
Limitations

You should be aware of some restrictions:

- Instances of sketch-based features cannot be moved apart from their prototype.
- Any parameter used as an argument in a formula should be preceded by the ? symbol. The syntax X = 2 * Y is invalid and should be replaced with X = 2 * **?** Y.
- Unless a formula-defined parameter has not been initialized with the proper units, the value calculated from the formula is dimensionless.
  Y = 0 kg ;
  Y = 2 * ? X ;
- A script error stops the reading and the execution of the loop.

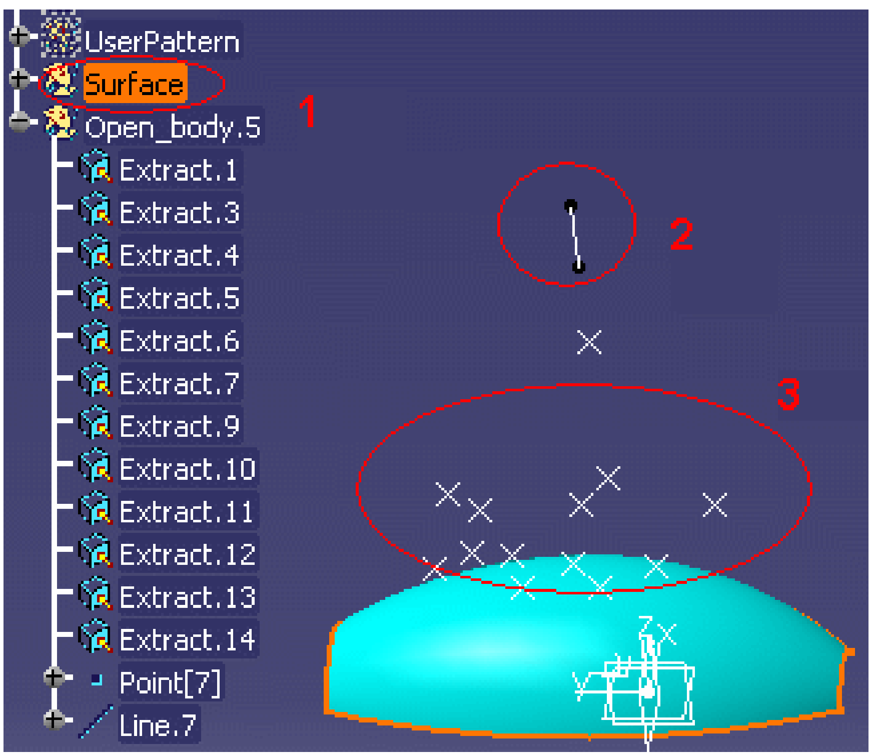
Creating a Loop

The task below illustrates how to interactively apply a loop to an existing document.



The KwrLoop1.CATPart is made up of a surface (**2**) and a solid (**1**) that symbolizes a hole. This hole is inserted into a User Feature (UDF) for a later instantiation. The User Feature (UDF) has 3 different inputs (a point, an axis and a surface). 2 parameters of the User Feature (UDF) are published (**3**).

- Clearance=4mm
- Diameter=8mm



The KwrLoop2.CATPart is made up of a surface (**1**) and of 12 points (**3**) inserted into a list. The Line.7 is the instantiation axis (**2**).

The aim of this scenario is to instantiate as many holes as existing points. It is divided into the following steps:

- The user creates a loop.
- The user instantiates the User Feature (UDF) from the existing .CATPart file.
- The user valuates the required inputs to instantiate the holes.

To create a loop, you have to:

1. Declare input data
2. Define the context
3. Specify iterators
4. Write the body of the action script

Before creating a loop in a CATPart document, make sure that the **Manual input** option is **unchecked** in the Part Number field of the **Tools->Options...->Infrastructure->Product Structure->Product Structure** tab.

To carry out this scenario, you will need the following files:

- KwrLoop1.CATPart
- KwrLoop2.CATPart

1. Open the KwrLoop2.CATPart

Creating a Loop

2. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench.
3. Click the **Loop** icon ( ) in the **Control Features** bar. The Loop Edition window displays.
4. In the specification tree, select the inputs of the loop.
   - Expand the Parameters node and select the List_Extract list. In the Input name field, enter the name of the list: PointsList.
   - Expand the Surface node and select the Revolute.1 feature. In the Input name field, enter the name of the list: SurfRef.

- Expand the Result_Body node and select the Line.9 feature. In the Input name field, enter the name of the list: LineAxis.

  Note that the name indicated in the Input name field is the one that will be used in the loop body.

5. Select the context, that is to say, in this scenario, the feature that will contain the instantiated holes.

   - Click the Context field.
   - Click Result_PartBody in the specification tree.

6. Indicate the number of holes that you want to instantiate into the surface.

   - In the From field, indicate 1.
   - Right-click the **To...** field and select the **Edit formula...** command. The Formula Editor displays.
   - In the specification tree, click ListSize=12. Click **OK** when done. The number of instantiated holes is now valuated by a formula based on the list, that is to say on the number of points contained in the list.

7. Enter the following action script into the Editor.

```
import "E:\users\Loops\KwrLoop1.CATPart";  1

UDF_$i$ isa Clearance_Hole_UDF  2
{
        Position = object: PointsList[$i$];
        Clearance_Surface = object: SurfRef;  3
        Axis = object: LineAxis;
}
```

- Use the import keyword to indicate the path of the file containing the User Feature (UDF) to be instantiated. To indicate the path of the file, it is recommended to use the Insert File Path command available in the contextual menu to import KwrLoop1.CATPart. (**1**)
- UDF_i is the name that will be attributed to each instance of the hole. At each iteration, i is replaced with the current iterator. (**2**)
- Clearance_Hole_UDF is the name assigned to the User Feature (UDF) in the KwrLoop1.CATPart file. (**2**)

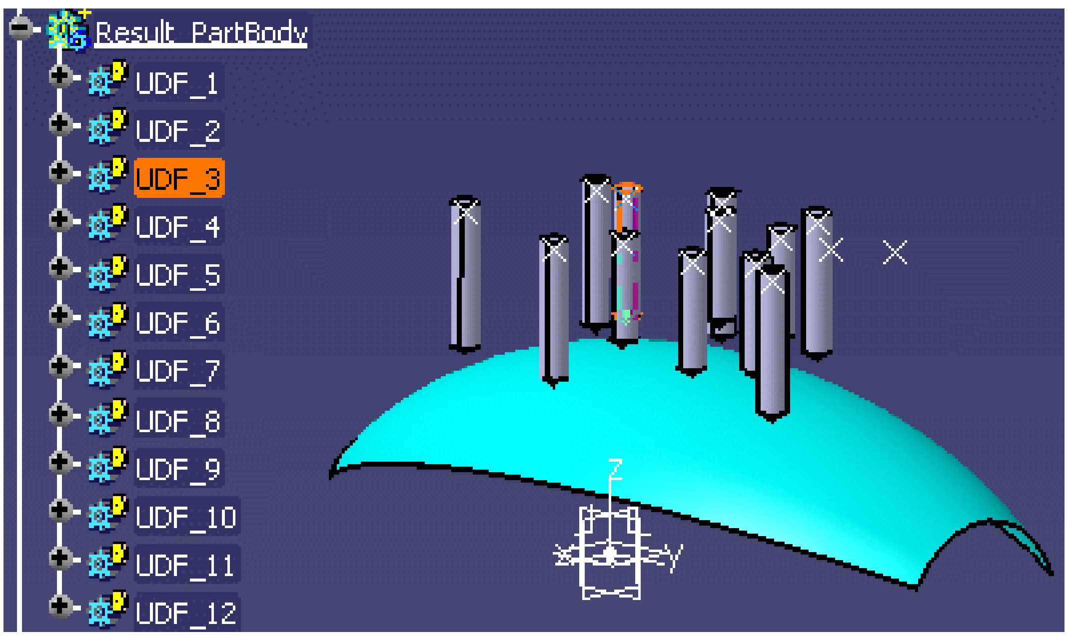
- Position is a point and is also the first input that needs to be valuated when instantiating the holes. PointsList is the name of the List. (**3**)

- Clearance_Surface is the second input required and defined when creating the User Feature (UDF) and SurfRef is the revolute into which the holes will be instantiated. (**3**)

- Axis is the third input required and defined when creating the User Feature (UDF) and LineAxis is Line.9, that is to say the instantiation axis. (**3**)

To know more about the syntax to be used (;, {}, i) in the loop body, see Using the Scripting Language.

8. Click **OK** when done. The 12 holes are instantiated. (See picture below.)

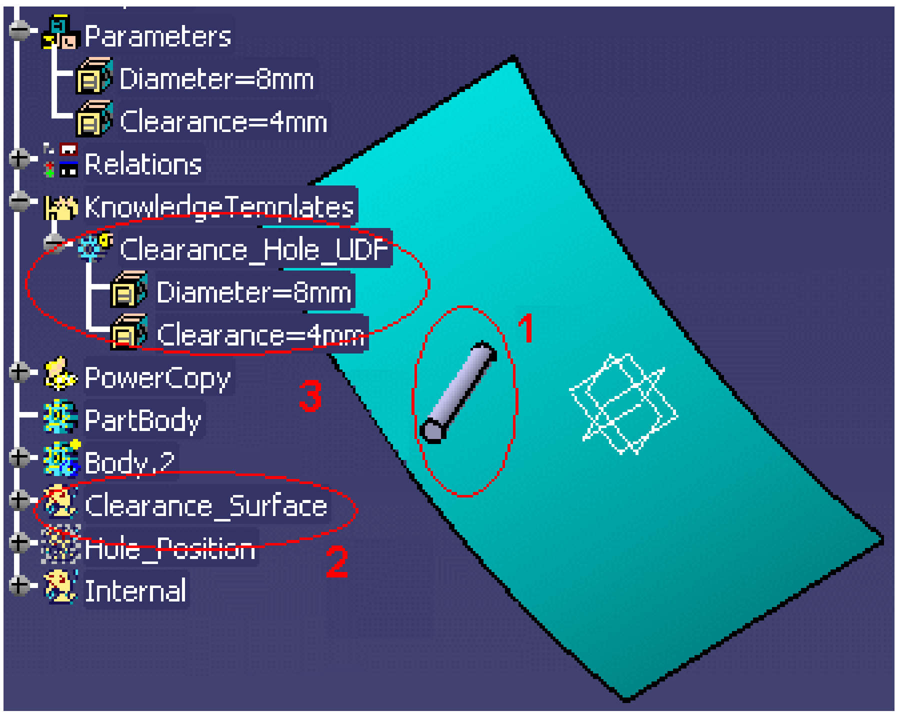


Creating a PowerCopy Containing a Loop

This task illustrates how to interactively apply a loop to an existing document. In this scenario, the user wants to make holes in a pad. To do so, he:

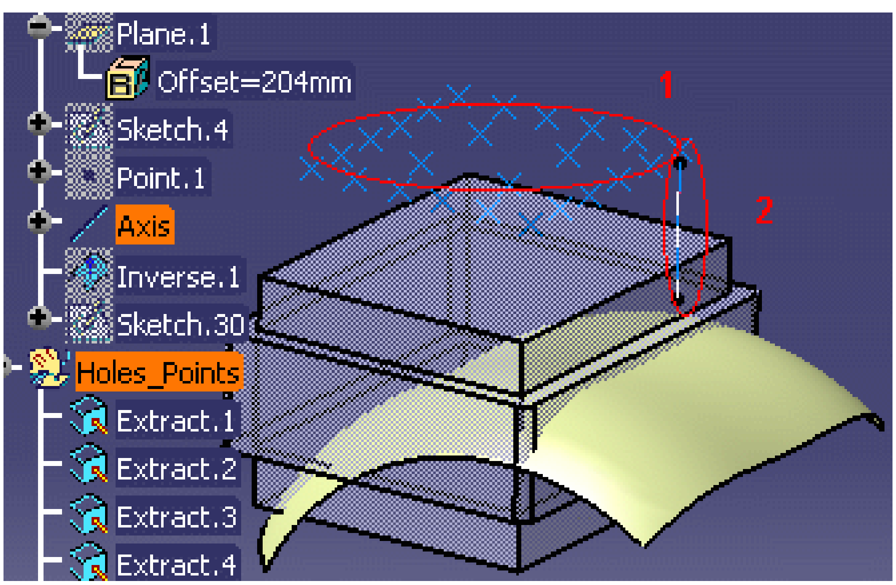
- Creates a loop referencing the inputs of an existing User Feature (UDF) used to make holes in a pad.
- Saves the loop in a powercopy.
- Instantiates the powercopy into an existing document and creates the holes.

To carry out the scenario, the user will need the following files:

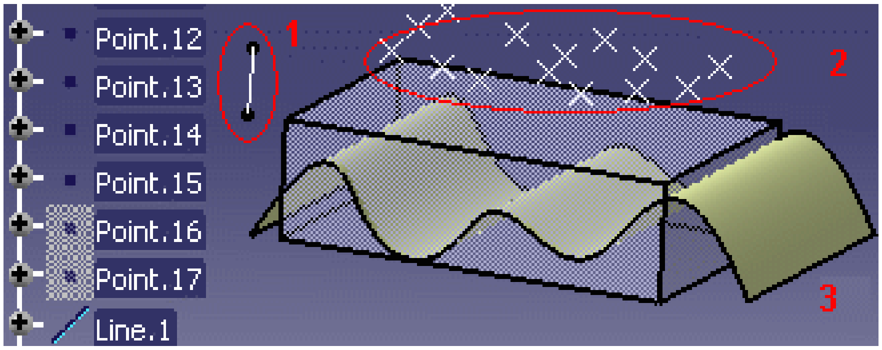


The KwrLoop1.CATPart is made up of a surface (2) and a solid (1) that symbolizes a hole. This hole is inserted into a User Feature (UDF) named Clearance_Hole_UDF for a later instantiation. The User Feature (UDF) has 3 different inputs (a point, an axis and a surface). 2 parameters of the User Feature (UDF) are published (3):

- Clearance=4mm
- Diameter=8mm



The KwrLoop3.CATPart file is made up of a pad and a surface and of 24 points (1) inserted into a list. Line.2 is the instantiation axis (2). This .CATPart file is the one that will contain the loop contained in the powercopy that will be instantiated into KwrLoop4.CATPart.



The KwrLoop4.CATPart is made up of a pad and a surface (**3**) and of 17 points (**2**) inserted into a list. Line.1 is the instantiation axis (**1**). It will contain the instantiated loop and the holes.

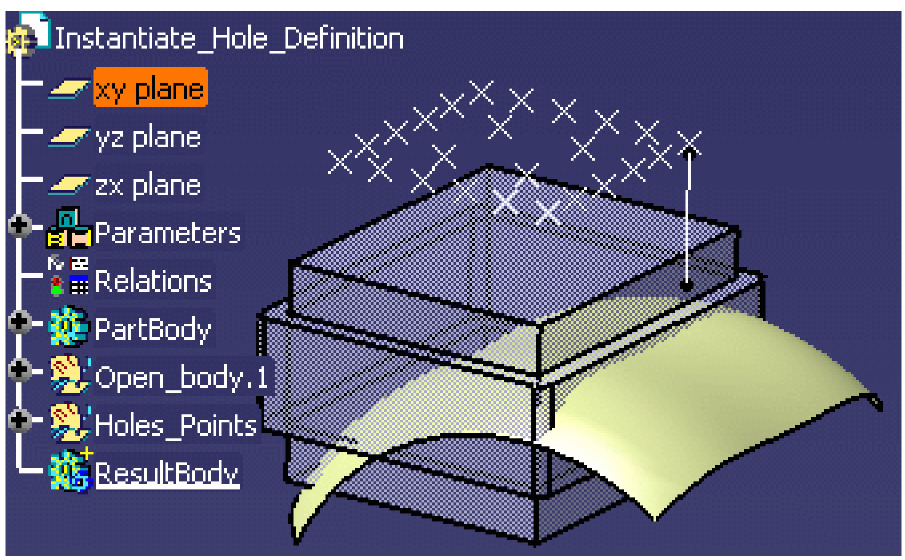
Before creating a loop in a CATPart document, make sure that the **Manual input** option is **unchecked** in the Part Number field of the **Tools->Options...->Infrastructure->Product Structure->Product Structure** tab.

To create a loop, you have to:

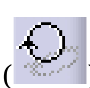
1. Declare input data
2. Define the context
3. Specify iterators
4. Write the body of the action script

Creating the loop referencing the user feature (UDF)

1. Open the KwrLoop3.CATPart. The following image displays.



2. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench.

3. Click the **Loop** icon () in the **Control Features** bar. The Loop Edition window displays.

4. In the specification tree, select the inputs of the loop.

   - Expand the Parameters node and click the **Lists_PointRef** list. In the Input Name field, enter the name of the list: PointsList.

   - Expand the Geometrical Set.1 node and select the Clearance_Surface feature. In the Input name field, enter the name of the feature: SurfRef.

   - Expand the Holes_Points node and select the Line.2 feature. In the Input name field, enter the name of the line: LineAxis.

   Note that the name indicated in the Input name field is the one that will be used in the loop body.

5. Select the context, that is to say, in this scenario, the feature that will contain the instantiated holes.

   - Click the Context field.

   - Click ResultBody in the specification tree.

6. Indicate the number of holes that you want to instantiate into the surface.

   - In the **From** field, indicate 1. (1 corresponds to Extract.1.)

   - Right-click the **To...** field and select the **Edit formula...** command. The Formula Editor displays.

- In the specification tree, click ListSize=24. Click **OK** when done. The number of instantiated holes is now valuated by a formula based on the list, that is to say on the number of points contained in the list.

7. Enter the following action script into the Editor.

```
import "E:\users\mei\R12\PES\kwr\loop\test\KwrLoop1.CATPart" ; 1

UDF_$i$ isa Clearance_Hole_UDF   2
{
                Position = object: PointsList[$i$];  3
                Clearance_Surface = object: SurfRef; 4
                Axis = object: LineAxis; 5
                Clearance = 3mm; 6
}
```

- Use the import keyword to indicate the path of the file containing the User Feature (UDF) to be instantiated (KwrLoop1.CATPart).

- To indicate the path of the file, it is recommended to use the Insert File Path command available in the contextual menu to import KwrLoop1.CATPart. (**1**)

- UDF_i is the name that will be attributed to each instance of the hole. (**2**)

- Clearance_Hole_UDF is the name assigned to the User Feature (UDF) in the KwrLoop1.CATPart file. (**2**)

- Position is a point and also the first input that needs to be valuated when instantiating the holes. PointsList[i] is the name of the List. [i] corresponds to the nth item of the list. In this case, nth is equal to 24, the number of holes to be instantiated (**3**).

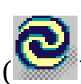
- Clearance_Surface is the second input required and defined when creating the User Feature (UDF) and SurfRef is the revolute into which the holes will be instantiated. (**4**)

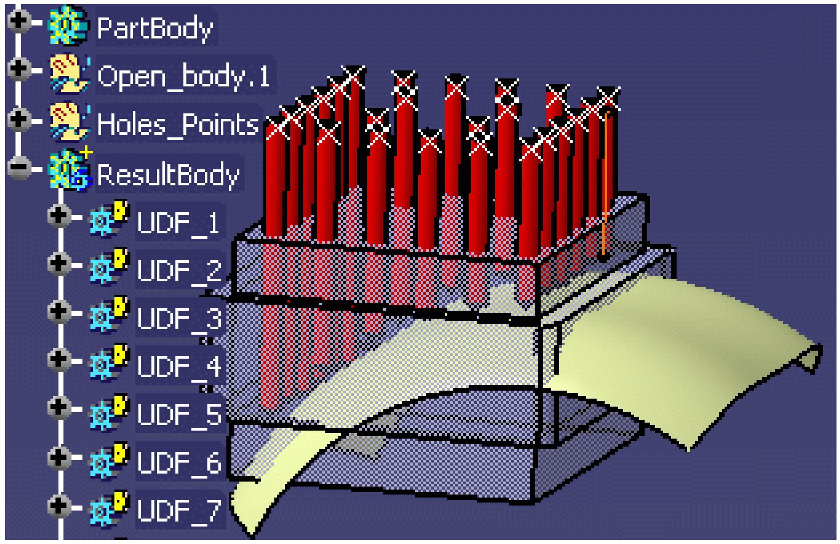
- Axis is the third input required and defined when creating the User Feature (UDF) and LineAxis is Line.9, that is to say the instantiation axis. (**5**)

- Clearance is one of the published parameters of the User Feature (UDF). It is used in the action script because the user wants the value of the published parameter to be modified. (**6**)

To know more about the syntax to be used (;, {}, i) in the loop body, see Using the Scripting Language.

8. Click **OK** when done. The holes are instantiated (see graphic below.)

9. Click the **Update** icon () to update the document.



10. Right-click the loop and use the **Properties** command to rename the loop into Loop_Holes. Click **OK** when done.

11. In the specification tree, right-click the loop (located below the Relations node) and select the **Loop_Holes object->Deactivate** command.

Saving the loop in a powercopy

12. Click the root of the specification tree, and from the **Start->Mechanical Design** menu, access the **Part Design** workbench.

13. From the **Insert->Advanced Replication Tools** menu, select the **PowerCopy Creation...** command. The Powercopy Definition window displays.

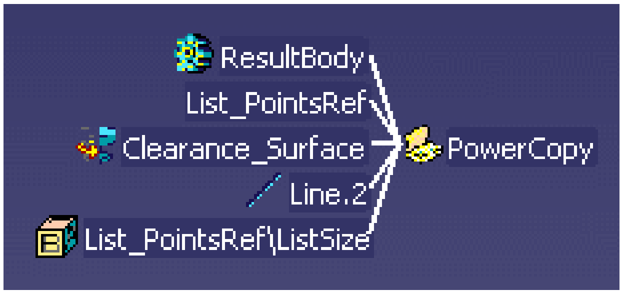
14. In the specification tree, select the items making up the powercopy:

    - Formula.1

    - Loop_Holes

Note that the powercopy will need the following inputs at instantiation time:

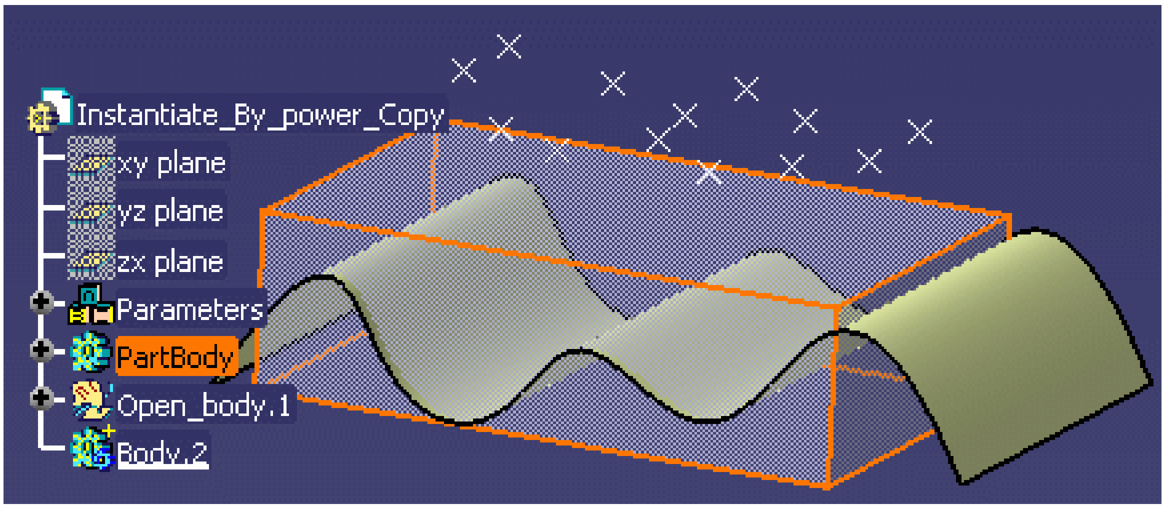
- ListSize
- Line.2
- Clearance_Surface
- List_PointsRef
- ResultBody



15. Click **OK** when done. The PowerCopy is created and displays below the PowerCopy node in the specification tree.
16. Save your file and close it.

Instantiating the powercopy into an existing document

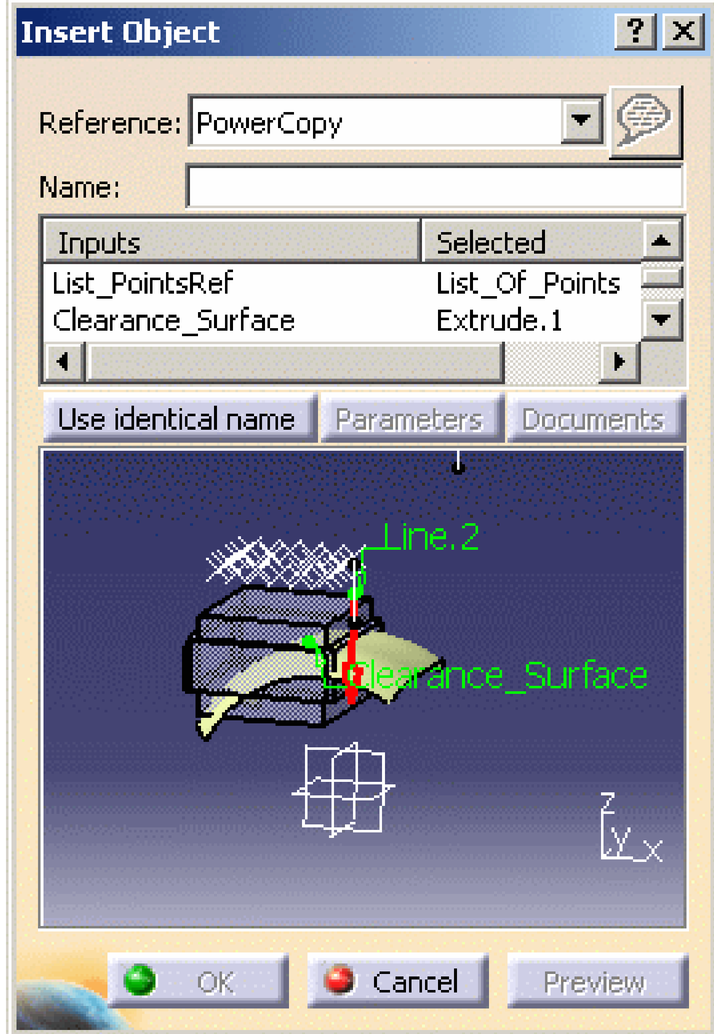
17. Open the KwrLoop4.CATPart file. The following image displays.



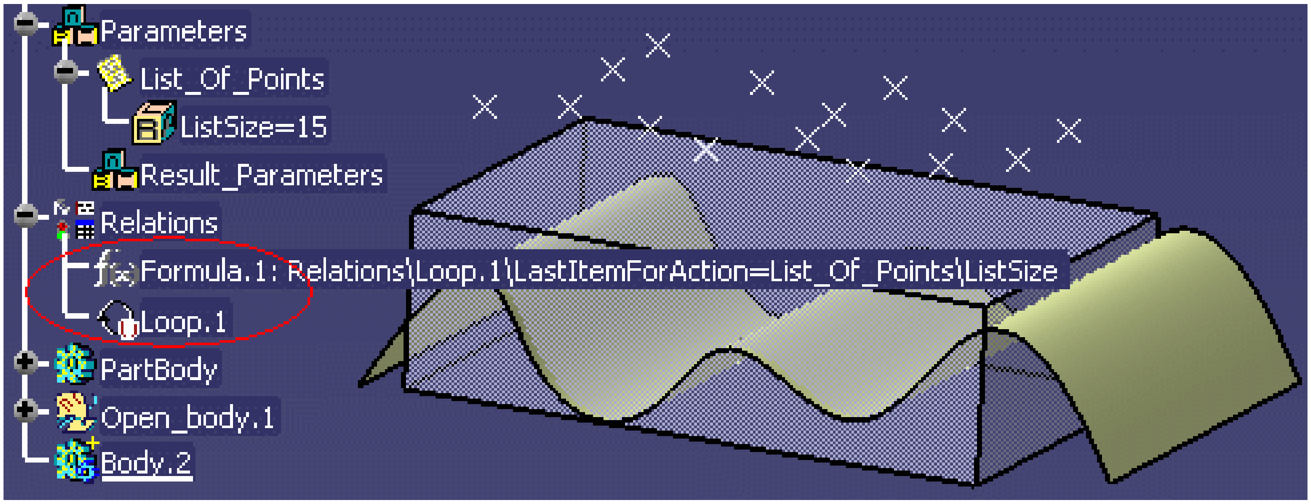
18. From the **Insert** menu, select the **Instantiate From Document...** command.
19. In the **File Selection** window, select the KwrLoop3.CATPart file that you have just saved and click **Open**. The Insert Object dialog box displays.

20. Valuate the inputs of the powercopy. To do so, in the specification tree, click:

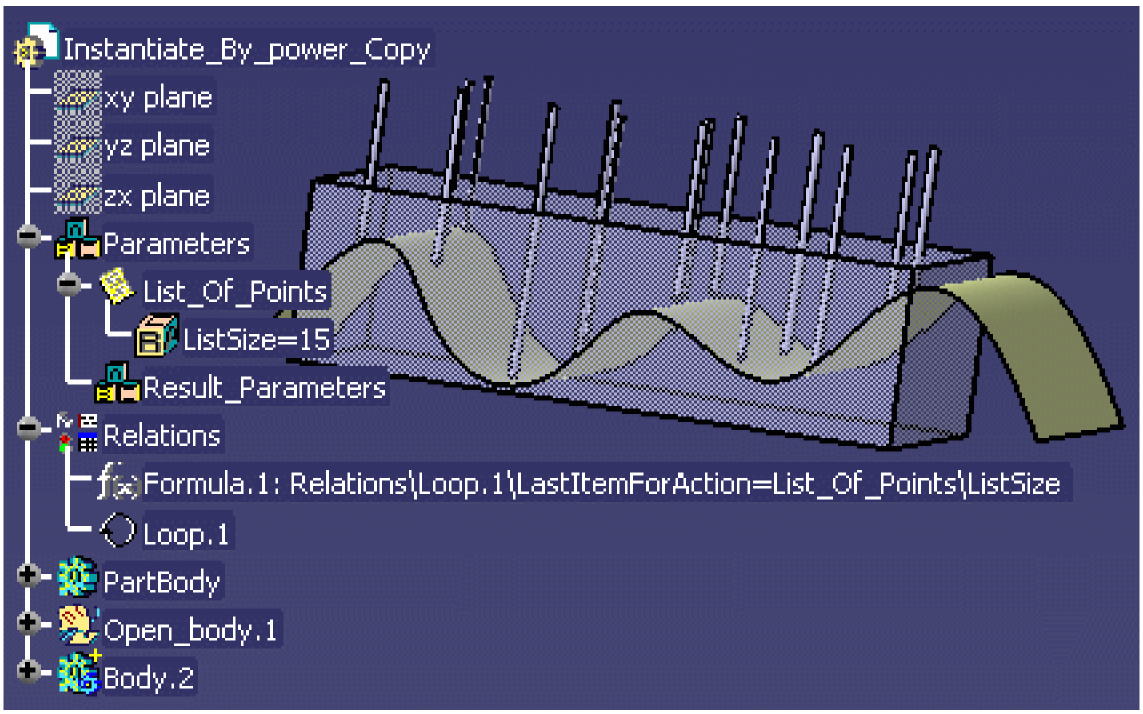
- ListSize=15 to valuate ListSize.
- Body.2 to valuate the ResultBody.
- List_Of_Points to valuate the List_PointsRef.
- Extrude.1 (located below Geometrical Set.1) to valuate the Clearance_Surface.
- Line.1 (located below Geometrical Set.1) to valuate Line.2, that is to say the instantiation axis.



21. Click **OK** when done. The Loop and the formula contained in the powercopy are instantiated.



22. To instantiate the holes, activate the loop. To do so, right-click Loop.1 in the specification tree and select the **Loop.1 object->Activate** command. The holes are instantiated.



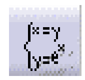
Solving a Set of Equations

This task explains how to solve a set of equations using the operators and functions of the knowledgeware language. This scenario can be run from any document.

- In a set of equations, the semi-colon (;) is used as a separator.
- Note that the equations set capabilities require the Knowledge Advisor product.

Note that a KWA license is required to execute loops.

1. Create two real type parameters x and y. Both parameters are intended to be used as variables in a set of equations.

2. Access the Knowledge Advisor workbench. Click the  icon. In the first dialog box which is displayed, enter the name of the relation, a comment and a destination. Then click **OK**. The Set Of Equations editor is displayed.

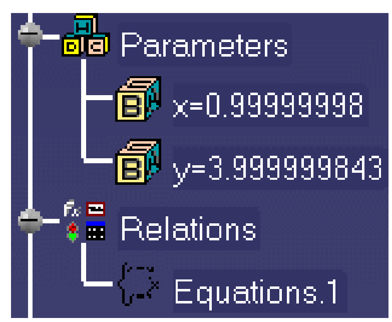
3. Enter the set of equations below into the edition box:

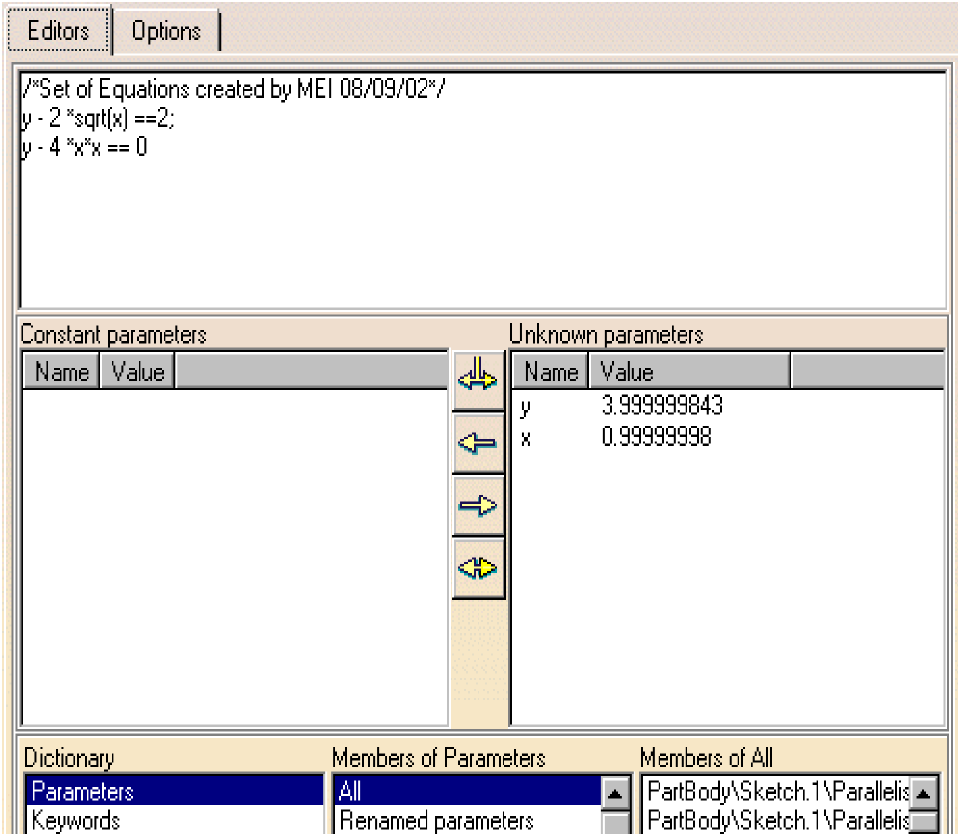
   y - 2 *sqrt(x) ==2;

   y - 4 *x*x == 0

   Now, your editor looks something like this:

The value of each parameter is displayed first in the Unknown parameters field, then in the specification tree (see below).





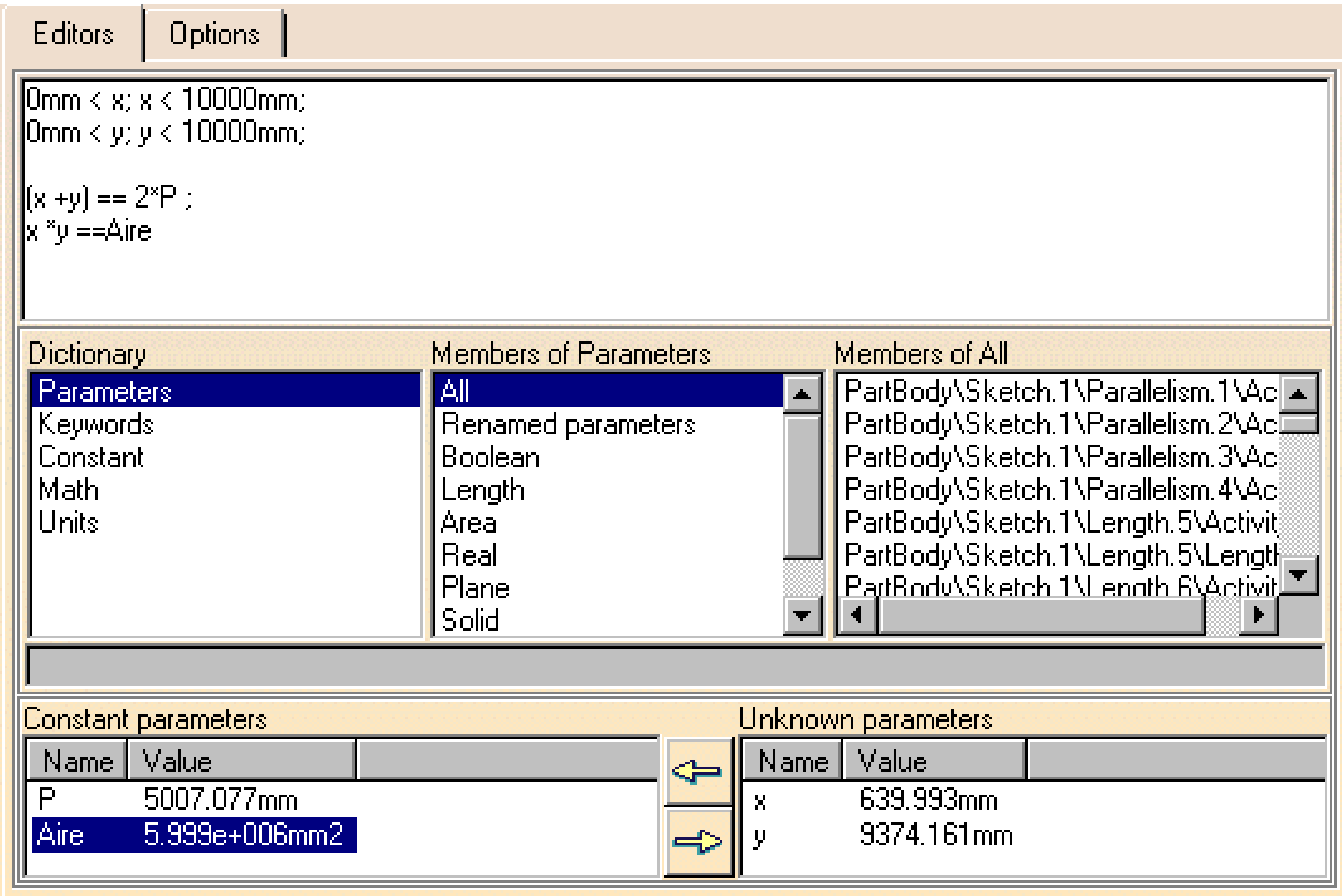
4. Click here to open the result file.

To know more about the Equations Editor, see Using the Equations Editor.

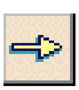
Using the Equations Editor

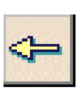
In order to improve the use of the Equations solving functions, the Equation Editor was modified. It is made up of two tabs: the Editors tab and the Options tab.

Editors tab




The Parse arrow is used to identify the variables of the set of constraints. It must be pushed before choosing input and output variables.


The left arrow is used to move variables from the Unknown parameters category to the Constant parameters one.

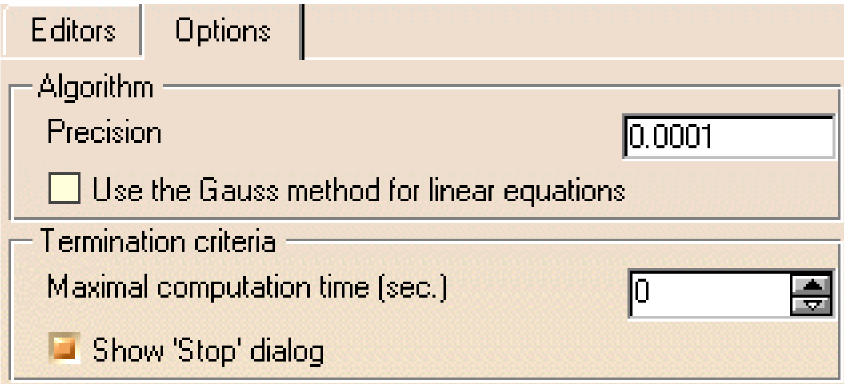

The right arrow is used to move variables from the Constant parameters category to the Unknown parameters one.


The Switch input/output arrow is used to swap the selected constant and unknown parameters.

- Viewer: enables you to enter the equations that you want to solve.
- Dictionary: see Using the Rule Editor.
- Members of Parameters: see Using the Dictionary.

- Members of All: see Using the Rule Editor.
- Constant parameters: Constant parameters: The value of constant parameters are set by the user and are considered as constants by the solver. This value can be changed directly in the Value column by clicking twice (slowly) in the Value cell.
- Unknown parameters: The value of unknown parameters will be calculated once the Apply button is pushed.

Options tab



Algorithm

- **Precision**: enables you to define the precision of the results (i.e the number of decimal digits after the decimal point.)
- **Use the Gauss method for linear equations**: accelerates the solve operation when working with linear equations.

Termination criteria

- **Maximal computation time (sec.)**: enables you to indicate the computation time. If the indicated time is equal to 0, the computation will last until a solution is found.
- **Show 'Stop' dialog**: if checked, displays a "Stop" dialog box that will enable you to interrupt the computation.

Solving a Set of Equations

Using the Knowledge Advisor Language

Writing Formulas

A formula is a one-line statement that you can write either by typing directly the appropriate syntax in the editor field or by selecting items from the editor dictionary list. The formula syntax is easy to use and learn.

The period is generally used as a separator between the whole numbers and the fractional part of a number. Using a comma as a separator in place of the period is not recommended in real values intended to be used directly in relations.
Example: Real1 = 2,1 + 5,4 is not allowed whereas Real1 = Real2 + Real3 is allowed regardless of the separator used when valuating Real2 and Real3.

Writing Rules and Checks

Rules and checks are **multi-line** statements that you can write either by typing directly the appropriate syntax in the editor field or by selecting items from the editor dictionary list. Here is a description of the syntax to be used. The mathematical and trigonometric functions as well as the functions used to manipulate strings are the same as for formulas.

Conditional Statements

Rules

if ... else ... else if

Conditionally executes a group of statements, depending on the value of an expression. You can use either block form syntaxes:

```
if condition statements [else elsestatements ]

or

if condition
   { statements }
[else if condition-n
   [ { elseifstatements } ] ] ...
[else
   [ { elsestatements } ] ]
```

You can use the single-line form (first syntax) for short, simple rules. However, the block form (second syntax) provides more structure and flexibility than the single-line form and is usually easier to read, maintain, and test.

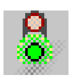
The **else** and **else if** clauses are both optional. You can have as many **else if** statements as you want below a block **if**, but none can appear after the **else** clause. Block **if** statements can be nested that is, contained within one another.

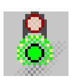
Checks

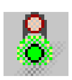
statement1 **=>** *statement2* (**if** *statement1* **then** *statement2*)

Displays a message (if type is Warning or Information) and turns to red in the specification tree each time *statement2* is invalid as *statement1* is fulfilled.

OK => KO 

KO => KO 

KO => OK 

OK => OK 

For Statement

Note that the For statement is available for Action and Reaction scripts only. To create loops, proceed as follows:

The first type of loop is a loop based on the element of a list. See syntax opposite.

```
For x inside List
{
... Body ...
}
```

Where:

- X is a variable name (of a given type. It may represent an object or a value).
- List is a variable name of type List or an expression returning a list.
- X (like any other variable of the language) can be used in the body. It contains the Nth element of the list.

The body is executed Nth times where N is the number of elements of the list.

The second type of loop executes until an expression becomes false. See syntax opposite.

```
For x while predicate
{
... Body ...
}
```

Where:

- X is a variable name of integer type. It is incremented at the end of each execution of the body.
- Predicate is a Boolean expression. The body is executed as long as this expression is true. This expression is evaluated before the body.

Note that the second for operator can lead to infinite loops.

While Statement

This loop executes until an expression becomes false. See syntax opposite.

```
let i = 1
let x(Point)

for i while i<=parameter.Size()
{
x = parameter.GetItem(i)
if (x.GetAttributeReal("Y") <
0.04)
x.SetAttributeReal("Y",0.04)
}
```

Where:

- i is a variable name of integer type. It is incremented at the end of each execution of the body.
- X is a variable for points.

Constants

The following constants are specified or recognized by *CATIA* when programming rules and checks. As a result, they can be used anywhere in a relation in place of the actual values.

- false - one of the two values that a parameter of type Boolean can have
- true - one of the two values that a parameter of type Boolean can have
- PI - **3.14159265358979323846** - The ratio of the circumference of a circle to its diameter.
- E - The base of natural logarithm - The constant *e* is approximately **2.718282**.

Comments

The **/*** and ***/** comment characters are supported.

```
/* Rule created by CRE 05/03/99 */
if PartBody\Sketch.1\Radius.3\Radius > 45mm
  {
   LaunchMacroFromFile("Macro1.CATScript")
  }
else
/*
   LaunchMacroFromFile("Macro2.CATScript")
*/
   Message("No macro launched")
```

Temporary Variables

Temporary variables can be declared by using the **let** keyword. A temporary variable does not persist as a parameter after the rule execution is finished.

```
/*Rule created by CRE 08/23/99*/
let x = 5 mm
if PartBody\Hole.1\Diameter > x
{
PartBody\Hole.1\Activity = false
}
```

For non digital values, the type has to be indicated:

```
let S(Surface)
S= split (...,...)
```

Temporary variables should be declared at the beginning of the rule, before any conditional instruction is specified.

```
let S1(Surface)
let S2(Surface)
let S3(Surface)

S1 = Split ...
S2 = ...
S3 = ...
```

Units

Units are all provided in the dictionary.

1. Pay attention to unit consistency when writing a rule or a check.
2. Units are written with an underscore instead of the usual "/" (example N_m2 instead of N/m2).

Operators

Arithmetic operators

+ Addition operator (also concatenates strings)

- Subtraction operator

* Multiplication operator

/ Division operator

( ) Parentheses (used to group operands in expressions)

= Assignment operator

** Exponentiation operator

Logical Operators

<**and** Logical conjunction on two expressions

or Logical disjunction on two expressions

Comparison Operators

<> Not equal to

== Equal to

>= Greater or equal to

<= Less than or equal to

< Less than

> Greater than

Object Methods

Description

Describes the parent of all mechanical features.

Attributes

ID	Owner
Name	NamedURLs
UserInfoComment	

Methods

AbsoluteId Method	AttributeType Method
GetAttributeInteger	GetAttributeBoolean
GetAttributeString	GetAttributeReal
ID Method	HasAttribute
IsOwnedBy Method	IsOwnedByString Method
IsSupporting Method	Name Method
Query Method	SetAttributeInteger
SetAttributeBoolean	SetAttributeReal
SetAttributeString	

Example

1. Create a part with several holes.

2. Add a real type parameter ("Real.1" for example) to one of the hole features. To do this, you must use the Knowledge Advisor product.
3. Create the rule below:

- List.1 is the name of the list on which the calculation will be performed.
- PartBody is the body on which the search will be carried out
- Hole is the Type.
- x.Diameter>50mm is the expression.

```
/* This rule resets the diameter of the hole */
/* which has "Real.1" as its parameter to the Real.1 value */
(for all) H:Hole
if H->HasAttribute("Real.1")
H.Diameter = 1mm*(H->GetAttributeReal("Real.1"))
```

You can use all the GetAttributexxx methods in that way.

- Add one or more drafts to the part.
- You can write the rule below:

```
(for all) Dr:Draft
/* Displays the names of the Drafts which have PartBody as their names */
```

Attributes

Id
Defines the feature identifier, i.e. the name primarily assigned to the feature at creation before any renaming has been done.

Owner
Defines the parent feature.

Name
Defines the feature name.

NamedURLs
Describes the URL that the user can add to a relation by clicking the Comment and URLs icon in the Knowledge Advisor workbench.

UserInfoComment
Describes the comment that the user can add in the Comment and URLs dialog box when adding a URL to a relation in the Knowledge Advisor workbench.

Methods

P2

AbsoluteId Method

Retrieves the path of a feature.

Syntax

feature.**AbsoluteId**(): String

Example

String.2=PartBody\Pad.1.Id() + PartBody\Pad.1.AbsoluteId()

Sample

KwrTopology.CATPart

AttributeType Method

Returns the attribute type in the form of a string.

GetAttributeBoolean Method

Returns the value of a boolean type parameter added to a given feature by using the Knowledge Advisor product. *parameterName* is the name of the boolean type parameter. It should be put between quotation marks (").This method enables to read:

- The attributes added to parameters using the Parameters Explorer.
- The real attributes added to objects.
- The User Properties of a product.

Syntax

feature.**GetAttributeBoolean(***String***):** Boolean

where the argument is name of the attribute.

Example

```
Message ("The value of the Boolean.1 attribute of # is #",
PartBody\Pad.1.Name(),
PartBody\Pad.1.GetAttributeBoolean("Boolean.1"))
```

Sample

KwrObject.CATPart

GetAttributeInteger Method

Returns the value of an integer type parameter added to a given feature by using the Knowledge Advisor product. *parameterName* is the name of the string type parameter. It should be put between quotation marks ("). This method enables to read:

- The attributes added to parameters using the Parameters Explorer.
- The real attributes added to objects.
- The User Properties of a product.

Syntax

feature.**GetAttributeInteger(***String***):** Integer

where *String* is name of the attribute. This name should be put between double-quotes.

Example

```
Integer.3=PartBody\Hole.1 .GetAttributeInteger("Integer.2")
```

Sample

KwrObject.CATPart

GetAttributeReal Method

Returns the value of a real or Length (in m) type parameter added to a given feature by using the Knowledge Advisor product. *parameterName* is the name of the string type parameter. It should be put between quotation marks ("). This method enables to read:

- The attributes added to parameters using the Parameters Explorer.
- The real attributes added to objects.
- The User Properties of a product.

Syntax

feature.**GetAttributeReal(***String***):** String

where *String* is name of the attribute. This name should be put between double-quotes.

GetAttributeString Method

Returns the value of a string type parameter added to a given feature by using the Knowledge Advisor product. *parameterName* is the name of the string type parameter. This method enables to read:

- The attributes added to parameters using the Parameters Explorer.
- The real attributes added to objects.
- The User Properties of a product.

Syntax

feature.**GetAttributeString(***String***):** String

where *String* is name of the attribute. This name should be put between double-quotes.

Example

```
String.2 =PartBody\Pad.1 .GetAttributeString("String.1")
```

Sample

KwrObject.CATPart

HasAttribute Method

Determines whether the attribute specified in the argument belongs to the feature the method is applied to.

Syntax

feature.**HasAttribute(***String***):** Boolean

where *String* is name of the attribute. This name should be put between double-quotes.

Example

```
Boolean.2 =
PartBody\Hole.1.HasAttribute("Real.1")
```

Sample

KwrObject.CATPart

Id Method

Applies to a feature. Retrieves the identifier of a feature.

Syntax

feature.**Id**(): String

Example

```
String.2=PartBody\Pad.1.Id() + PartBody\Pad.1.AbsoluteId()
```

Sample

KwrTopology.CATPart

IsOwnedBy Method

Determines whether the feature specified in the argument is the parent of the feature the method is applied to. *featureName* should be put between quotation marks (").

Syntax

feature.**IsOwnedBy**(): Boolean

Example

```
Boolean.1=PartBody\Hole.1.IsOwnedBy(PartBody)
```

Sample

Topology.CATPart

IsOwnedByString Method

Applies to a feature. Determines whether a feature belongs to another. This method returns a string.

Syntax

feature.**IsOwnedByString**(): Boolean

IsSupporting

Function indicating if the object passed in argument is supported or not.

Example

```
H:Hole
H->IsSupporting("TaperedHole") == true
```

Name Method

Applies to a feature. Retrieves the name of a feature. Cannot be used to rename a feature.

Syntax

feature.**Name**(): String

Example

```
String.1=PartBody\Pad.1.Name()
```

Sample

KwrTopology.CATPart

Query

Query()
Function used to search for the features located below the feature to which it applies and that verifies the specified expression and that adds these features to the list.
In the example below, the result of the search will return the holes of PartBody whose diameters are greater than 50mm.
Example: List.1=PartBody.Query("Hole","x.Diameter>50mm")
Where:

- List.1 is the name of the list on which the calculation will be performed.
- PartBody is the body on which the search will be carried out
- Hole is the Type of the searched feature.
- x.Diameter>50mm is the expression.

SetAttributeBoolean Method

Assigns the value specified in the second argument to the parameter whose name is specified in the first argument. *parameterName* is the name of the boolean type parameter whose value is to be modified. It should be put between quotation marks ("). *booleanvalue* is either TRUE or FALSE.

Syntax

feature.**SetAttributeBoolean(***String*, *Boolean***):** Void

where the first argument is name of the attribute while the second is the value to be assigned to it.

Example

```
if PartBody\Pad.1\Boolean.1 <> true
PartBody\Pad.1.SetAttributeBoolean("Boolean.1", true)
```

Sample

KwrObject.CATPart

SetAttributeReal Method

Assigns the value specified in the second argument to the parameter whose name is specified in the first argument. *parameterName* is the name of the real type parameter whose value is to be modified. *parameterName* should be put between quotation marks (").

Syntax

feature.**SetAttributeReal(***String*, *Real***):** Void

where *String* is name of the attribute and *Real* the value to be assigned to the parameter.

Example

```
if PartBody\Hole.1\Real.1 <> 3
PartBody\Hole.1 .SetAttributeReal("Real.1",3)
```

Sample

KwrObject.CATPart

SetAttributeString Method

Assigns the value specified in the second argument to the parameter whose name is specified in the first argument. *parameterName* is the name of the string type parameter whose value is to be modified. *parameterName* and *stringvalue* should be put between quotation marks (").

Syntax

feature.**SetAttributeString(***String*, *String***):** Void

where the first argument is name of the attribute while the second is the value to be assigned to it.

Example

```
if PartBody\Pad.1.GetAttributeString("String.1") <> "String1"
PartBody\Pad.1 .SetAttributeString("String.1","This is a test")
```

Another syntax for the same rule is:

```
if PartBody\Pad.1\String.1 <> "String1"
PartBody\Pad.1.SetAttributeString("String.1","This is a test")
```

Sample

KwrObject.CATPart

SetAttributeInteger Method

Assigns the value specified in the second argument to the parameter whose name is specified in the first argument. *parameterName* is the name of the integer type parameter whose value is to be modified. *parameterName* should be put between quotation marks (").

Syntax

feature.**SetAttributeInteger(***String*, *Integer***):** Void

where the first argument is name of the attribute while the second is the value to be assigned to it.

Example

```
if PartBody\Hole.1\Integer.1 <> 3
PartBody\Hole.1 .SetAttributeInteger("Integer.1", 3)
```

Sample

KwrObject.CATPart

Messages and macros

LaunchMacroFromDoc Function

Executes a macro stored in a document from a rule.
A macro is stored in a document when you don't specify any external file before recording it.

Warning: It is up to the user to check that the macro which is run is not going to cause an infinite loop or result in a system crash.

Syntax

LaunchMacroFromDoc(*MacroName*)

Example

LaunchMacroFromDoc("Macro1")

Question Function

Displays a message in a dialog box, waits for the user to click a button and returns a value indicating which button the user clicked (true if Yes was clicked, false if No was clicked)

Syntax

Question(String [# String1 # String2 ..., Param1Name, Param2Name, ...]): Boolean

The **Question** function takes one required argument and several optional arguments depending on whether parameter values are to be displayed in the message.

Arguments	***Description***
String	Required. String to be displayed in the dialog box (should be put in quotes).

# String1, Param1Name...	Optional. When parameter values are to be displayed within the message, the arguments should be specified as follows: • one string in quotes including a # symbol wherever a parameter value is to be displayed • as many [, parameter name] statements as parameter values declared with a "#" in the message.

Use the "\|" symbol to insert a carriage return in a prompt.

Example

```
Boolean2 =
Question("SketchRadius is # | Do you want to change this value ?",
PartBody\Sketch.1\Radius.3\Radius )
```

Note that you can use the Question function together with the BuildMessageNLS function for your question to display in your language. To use this function, use the following syntax:

question(BuildMessageNLS ("x","xx",a,b))

- x corresponds to the name of the CATXXX.CATNls file where you will find the NLS message (it is the CATXXX name without the CATNls extension).
- xx corresponds to the key name in this catalog.
- a and b are the arguments (values that will be replaced in the message)

LaunchMacrofromFile Function

Executes a macro CATScript from a rule.

Warning: It is up to the user to check that the macro which is run is not going to cause an infinite loop or result in a system crash.

Syntax

```
LaunchMacroFromFile("MacroName.CATScript" )
```

Example

```
LaunchMacroFromFile("Macro1.CATScript")
```

Run Method

Runs a macro with arguments.

Warning: It is up to the user to check that the macro which is run is not going to cause an infinite loop or result in a system crash.

Syntax

VB Script.**Run(***valueOrFeature:ObjectType,...***):** Void

where *valueOrFeature* is the macro argument name. There can be several arguments.

Example

You must have created the VB Script.1 macro prior to creating the rule below:

```
if PartBody\Pad.1.HasAttribute("String.1") == true
`VB Script.1` .Run(PartBody\Pad.1 .GetAttributeString("String.1"),PartBody\Pad.1.Name() )
```

Sample

KwrObject.CATPart

Message Function

Displays a message in an information box. The message can include one or more parameter values.

Syntax

Message(String [# String1 # String2 ..., Param1Name, Param2Name, ...]) : Void

The **Message** function takes one required argument and several optional arguments depending on whether parameter values are to be displayed in the message.

Arguments	***Description***
String	Required. String to be displayed in the information box (should be put in quotes).
# String1, Param1Name...	Optional. When parameter values are to be displayed within the message, the arguments should be specified as follows: • one string in quotes including a # symbol wherever a parameter value is to be displayed • as many [, parameter name] statements as parameter values declared with a "#" in the message.

Use the "|" symbol to insert a carriage return in a message.

Example 1

```
Message("External radius is: # | Internal Radius is: #",
PartBody\Sketch.1\Radius.3\Radius,
PartBody\Hole.1\Diameter)
```

Example 2

Note that this function can be used along with the buildMessageNLS function

```
Message (BuildMessageNLS("KwrCATCatalog.CATNls","Zero"))
```

Where x,y,z are parameters.

Note that you can use the Message function together with the BuildMessageNLS function for your question to display in your language. To use this function, use the following syntax:

Message(BuildMessageNLS ("x","xx",a,b))

- x corresponds to the name of the CATXXX.CATNls file where you will find the NLS message (it is the CATXXX name without the CATNls extension).
- xx corresponds to the key name in this catalog.
- a and b are the arguments (values that will be replaced in the message)

Limitations

Parameters and Formulas

- The output parameter (the one that is valuated) of a formula can not be a publication. The bypass is to use a rule.
- If you want to edit a formula located in another document (not the UI active one) in the Formula editor, a warning is displayed and the formula editor is grayed out.

Design Tables

When working with design tables in .xls format, the **Undo** command may not work properly.

Rules

- Geometrical features valuated by a rule should not be used as the construction support of sub-elements.
- If you create a rule driving formulas activity, you might encounter update problems.

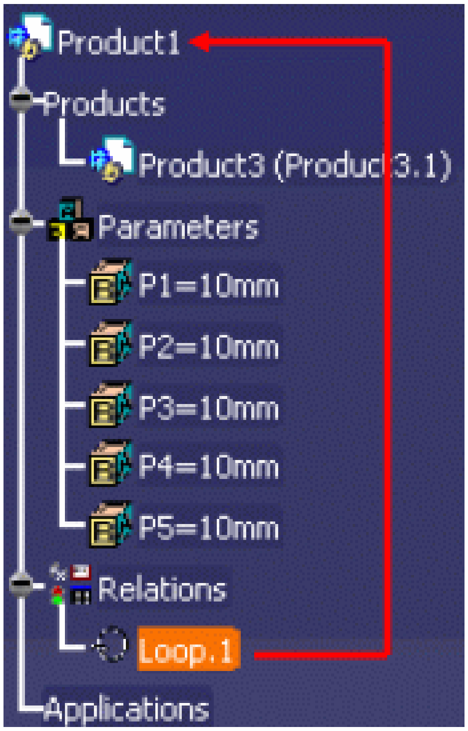
Sets of Equations

In set of equations, there is no control of units coherence (like in rule, check, formula).

Macros with Arguments

It is not possible to use the InputBox Visual Basic function in the definition of macros with arguments on unix. The SelectElement method on Selection object does not work either (on all Operating Systems).

Loops



A loop can only generate items in the document it belongs to. So, in the graphic opposite, the loop cannot generate items in Product3.

this restriction applies also to assemblies made up of different parts. A loop located in one of the Parts cannot generate items in another Part of the assembly.

Useful Tips

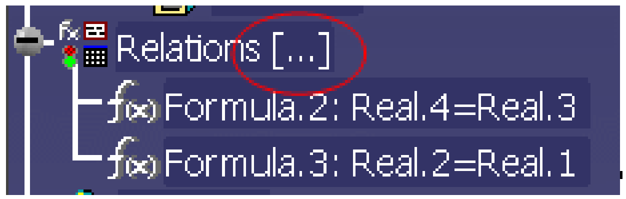
Relations

When using some objects, you need to indicate the destination of the formulas and the rules that valuate the parameters of these objects. If you valuate a time parameter in a kinematic simulation for example, the relation will not be located below the Relations set but in the mechanisms and commands tree of the simulation.

Hiding relations

You can hide a knowledge relation (formula, rule, check, ..) by right-clicking this relation in the specification tree and by selecting the **Hide** command.

- A visual indicator located at the Relations set level indicates that the set contains hidden relations.
  Note that this indicator is not recursive.
- If the user tries to delete a relations set containing hidden relations, a message displays asking the user if he wants to delete the relations set that contains hidden relations.
  Note that if the user tries to delete a relations set containing another relations set with hidden relations, the message will display.



Relations Updates

The evaluation of relations containing measures can be integrated to the Part update only. In a .CATProduct or in a .CATProcess file, to create a parameter

- which value is the result of a relation containing measures
- updated when modifying the measure inputs, proceed as follows:

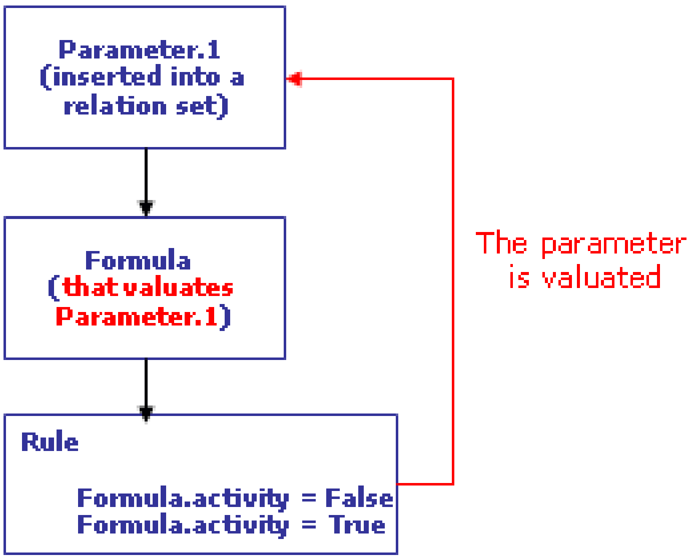
1. Create the relation containing the measure at the Part level.
2. Integrate the relation evaluation to the Part update.
3. At the Product or the Process level, create a relation that valuates the parameter by using the result parameter of the relation created at the Part level. To get an example, see KwrUpdate.CATProduct.
4. Perform a local update at the Relations level.

Rules

Geometrical Features and Rules

In a rule using features that need the geometry to return the type (such as extrudes), when the feature is deactivated, the type cannot be returned. To solve the problem, use the Set command to indicate the type in the rule. To know more, see the KwrSetType.CATPart file.

Rules and Update Cycle



This configuration is allowed since a modification of the parameter activity does not impact the formula update. But:

- In such a case, it is highly recommended to use the reaction feature.
- If you want to use a rule, do not deactivate and reactivate the activity parameter.
- When working with a UDF feature, make sure that you have inserted the relation set when defining the UDF.

Parameters

- You can add properties to a .CATPart or a .CATProduct document by using the Properties command from the contextual menu. You just have to click the Define other properties... button in the Product tab then click New parameter of type. The dialog is similar to the *f(x)* dialog. See the *Product Structure User's Guide* for more information. The properties you define that way are also displayed in the parameter list of the *f(x)* dialog box.

- Parameters belonging to a parameter set can be reordered by using the Reorder... command from the contextual menu.

- Parameters added by using the Parameters Explorer are displayed right below the feature they are assigned.

- *CATIA* users working with non-latin characters should check the **Tools->Options>Knowledge->Parameter Names->Surrounded by**'option. Otherwise, parameter names should have to be renamed in latin characters when used in formulas.

- You can specify that a parameter is constant by using the Properties command from the contextual menu. This command also enables you to hide a parameter.

- When copying parameters sets containing hidden parameters, these parameters are automatically pasted when pasting the parameters sets and appear as hidden parameters.

- Parameters have 2 different names: The local one and the global one.
  - The local name is the name attributed to the parameter when it was created in the Formula Editor or in the Parameters Explorer. Note that this name will not be modified if you perform a Reorder using the contextual menu. The local name can only be modified using the Parameters Explorer.
  - The global name (name) is the name attributed to the parameter by Knowledge Advisor. It is the path of the parameter + its type. If you select the parameter and reorder it, the path contained in the name will be modified. If you double-click the parameter in the specification tree, and enter a new name in the Edit Parameter dialog box, the global name will be changed. If, after renaming the parameter in the Edit Parameter dialog box, you reorder the parameter the path will not appear any more.

- Deleting parameters used in a relation: If you delete a parameter used in a relation, a "clone" parameter will be created.

- Applying the same formula to several parameters: If you want to apply the same formula to several parameters, use the Equivalent Dimensions feature and value this feature by a formula. To know more, see Using the Equivalent Dimensions Feature.

Formulas

- The Incremental option of the formula editor :The Incremental option allows you to restrict the list of parameters displayed in the dictionary. Select a feature either in the tree or in the geometry area. Only the first level of objects right below the selected feature will be displayed in the dictionary. If the Incremental option is unchecked, all the objects below the selected feature are displayed.
  The Incremental mode is useful when you work with large documents and when the parameter lists are long.

- Tips about the formula editor: To help you write a formula, the formula editor provides you with a dictionary. This dictionary exposes the list of parameters and functions you can use to define a formula. Depending on the category of objects to be referred to in the formula, the dictionary is divided into two or three parts. To insert any definition in the formula editor, just double-click the object either in the dictionary or in the tree. If you double-click a function in the dictionary, its signature is carried forward to the formula editor. Only the argument definitions are missing.

Design Tables

- A design table can only be created from *non-constrained parameters*, i.e. from parameters which are neither referred to in an active design table nor used in any other *active relation*.
  If you keep the Activity option checked for DesignTable0 and you try to create another design table, you will have to select the parameters to add to your second design table among a restricted parameter list.
  Uncheck the Activity option if you want to deactivate a design table and reuse its parameters in another design table.

- Anytime you modify a design table, the relations that refer to this design table detect the modification and turn to a to-be-updated status.

- As long as a design table is active, the parameters which are declared in it are constrained parameters and you are not allowed to modify them.
  Double-clicking a design table in the specification tree displays the design table with its set of configurations and allows you to select a new configuration.

- Only parameters which are not already constrained by any other relation or by any other design table can be used to create a design table. If a parameter is already constrained, it does not appear in the Parameters to insert list in the design table dialog box.

- **Selecting the parameters to be inserted in a design table**
  The Filter Name and Filter Type filters can be used to restrict the display of a parameter list. If you specify x in the Filter Name field of the Select parameters to insert dialog box, you will display all the parameters with the letter x in their name (xA, xB, xC, xD, xE). If you select the Renamed Parameters in the Filter Type list, you will display all the parameters you have renamed in the Formulas dialog box (yA, xB, xA, yC, xC, yB, yD, xD, yE, xE, TangE).
  Parameters to be inserted can be multi-selected. You just have to keep on pressing the Ctrl key while you select parameters. If you do this, the group of multi-selected parameters will be carried forward onto the Inserted parameters list in the order in which they are displayed in the initial list.
  When the design table is created, the rank of the columns fits the rank of the parameters in the Inserted parameters list. If you want to have columns ordered in a given way in the design table, you must insert the parameters one by one.

- **Accessing the functions related to the design table**
  Once in the formula (rule or check) editor, select the Design Table item in the dictionary, the list of the methods that can be applied to a design table is displayed. Select a method, then click F1 to display the associated documentation.

Loops

- Generic Naming: Generic naming is a *CATIA* technique which creates a label whenever an element has been selected interactively. This label is a coded description of the selected element. When you specify a fillet to be applied to a face, you must select interactively the face definition but prior to doing this you must of course have generated the face to be filleted. This is why scripts requiring face, point or edge definitions cannot be generated in one shot. You don't have to mind about the generic naming itself as it is automatically captured from the geometry area. The thing you have to mind about is the order your instructions are to be written and executed in the script.

- Message "*property* does not exist...": Check in the browser that the attribute name is correct. For attributes of list type (Fillets and Chamfers), check the indexes. The indexes specified must be consecutive from 1 to n without any gaps.

- Specifying a File Path (2 methods)

Syntax

There are 2 ways to specify the file path:

- **import "File path";** : Indicate the path of the file to be imported: `import "E:\users\kwecx\Models\PartImport.CATPart" ;`. Note that you should enclose the path within quotation marks and end the import statement with a semicolon (;).
- **import "File Name";** : Indicate only the name of the file to be imported if this file is located in the same directory as the document containing the loop: `import "PartImport.CATPart";`. Note that:
  - You should enclose the document name within quotation marks and end the import statement with a semicolon (;).
  - The file to be imported should be located in the same directory as the document containing the loop.
  - The document containing the loop should be saved.

To specify a file to be imported, you can use the 'Insert File Path' command from the contextual menu. Selecting this command displays a file selection panel. Quotation marks are automatically included but not the semicolon

- Importing Sketches: Recommendation

When designing a document to be generated by a script, it is better to group all the required sketches in a single file. That way:

- you minimize the overall size of your sketch-related data
- no matter the method used to specify the input file, you just have to specify the path once
- the design of the final document is made easier. You get a global view of the sketches on which the other features rely.

- Specifying Strings: Recommendation: Double quotation marks as well as single quotation marks of **apostrophe** type (`) can be used to delimit strings. Single quotations marks (`) must be used to enclose character strings which contain other strings.

Reactions

There are 2 ways to react to parameter changes using the reaction feature:

- Select the parameter and react to its Value Changed event

Or

- Select the feature (that owns the attribute corresponding to your parameter) and react to the Attribute Modification event. Note that this method does not work in all cases.

Use Cases

The Ball Bearing

A bearing is defined by parameters such as its principal dimensions, its basic load ratings, its limiting speeds and its mass. It belongs to a category which corresponds a certain range of its parameter values. In a catalogue, a bearing is referred to by a designation. Bearing types are described by tables which define the bearing parameter values including the designation.

The bearing example has been chosen here because the bearing tables given in distributor and retailer catalogues illustrate quite well the design table principles. The bearing itself is a good example of how components within a mechanical part can be constrained by relations.

In the scenario below, you start from an existing document inspired by a deep groove ball bearing. This document contains already a number knowledgeware relations, others are added to control the document design.

Before you Start
Step-by-Step

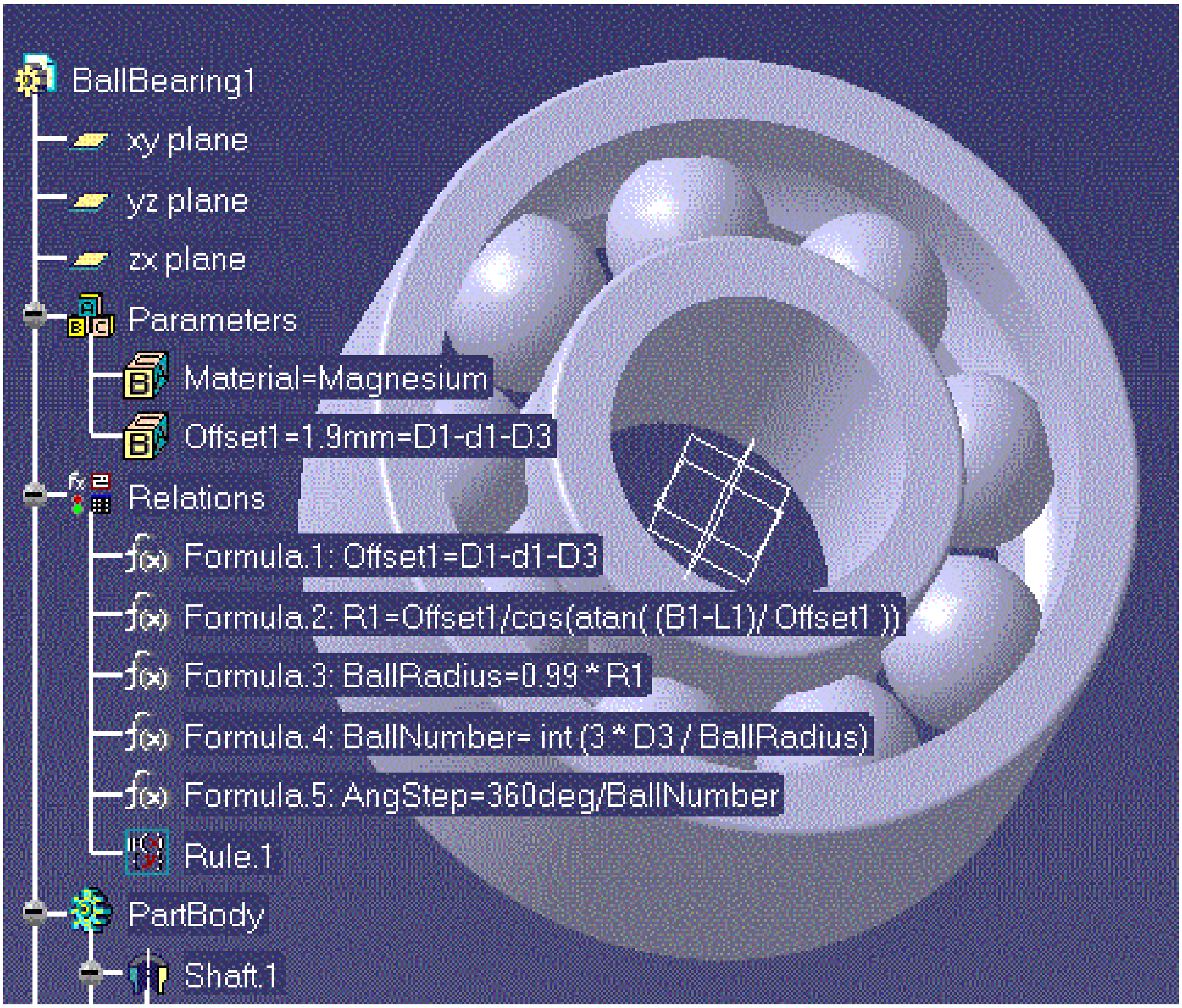
Before you Start

Here is the data required to carry out the scenario. They are all delivered with the Knowledge Advisor product but can be rebuilt from the information given below.

See the *Infrastructure User's Guide* for how to specify the material library settings (you must use the **Tools->Options...->Infrastructure->Material Library** command from the standard menu bar).

The Initial Document

The initial document is the KwrBallBearing1.CATPart document.

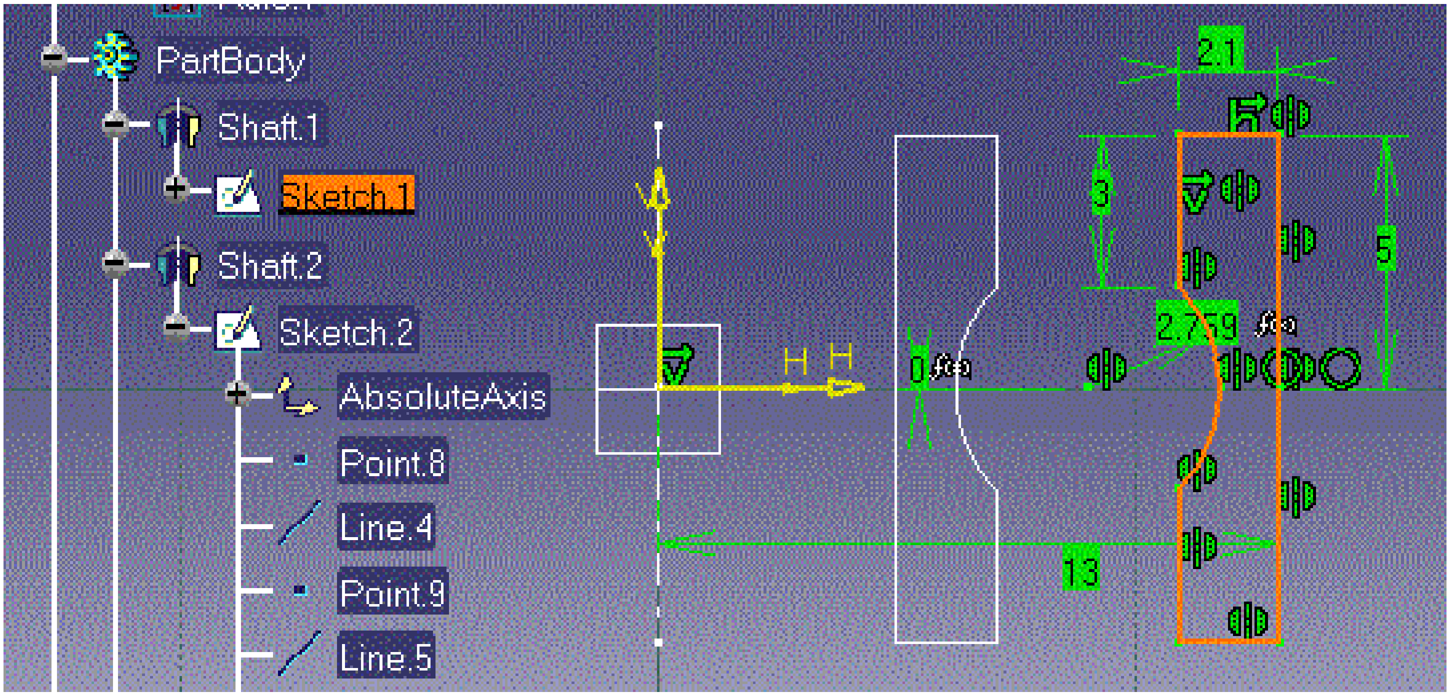


The bearing rings are coaxial shafts created from the Sketch.1 and Sketch.2 features. The balls are shafts created from the Sketch.3 feature.

The Outer Ring

The outer ring is a shaft generated by rotating the Sketch.1 highlighted in figure below around an axis coaxial to

V. Note that you must create this axis as a *construction element*, otherwise *CATIA* won't let you create the Shaft. The lower part of the sketch is the symmetry of the upper part with respect to the H axis.

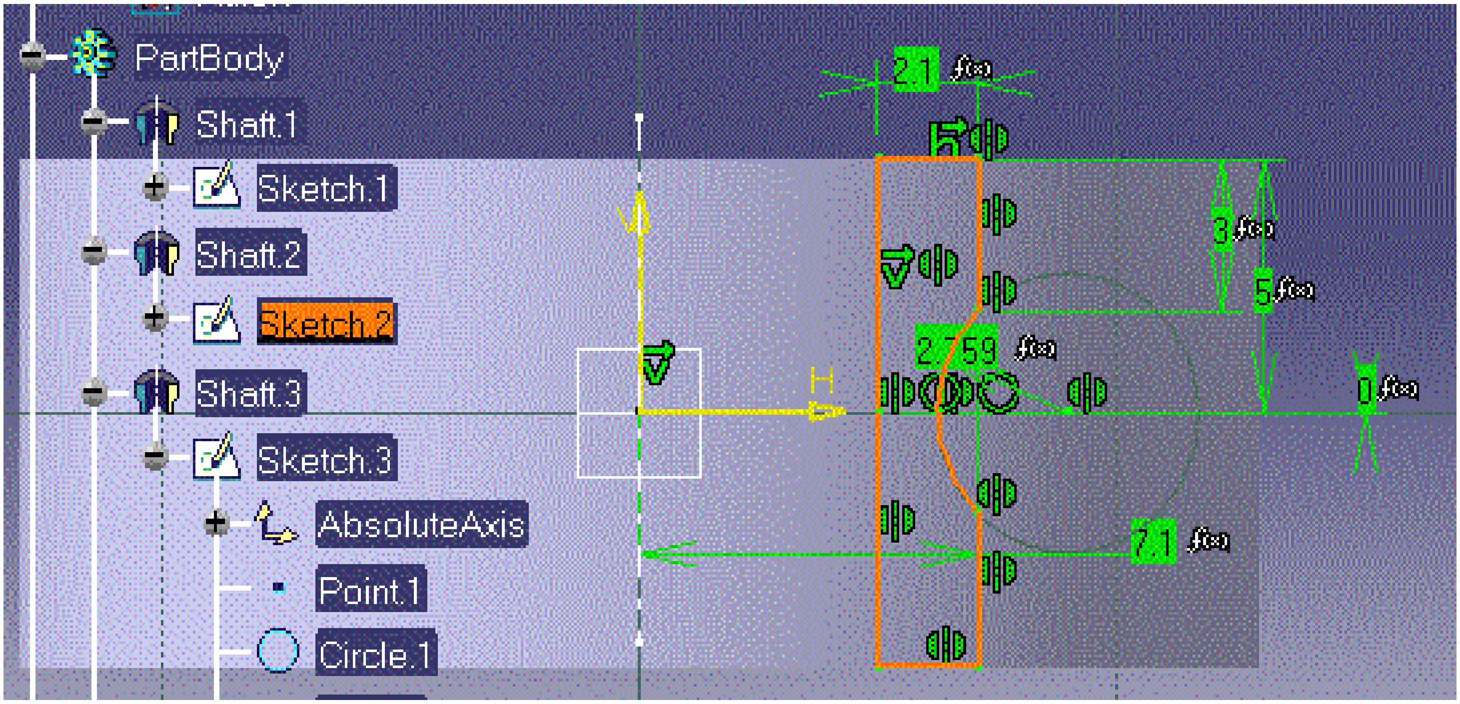


Here are the constraints defined on this sketch:

d1	2.1 mm	*ring width*
L1	3 mm	*half height of the non - hollowed inner surface*
B1	5 mm	*half height of the outer surface*
R1	2.759 mm	*groove radius*
b1	0 mm	*ordinate of the groove center*
D1	13 mm	*external diameter*

The Inner Ring

The inner ring is a shaft generated by rotating the Sketch.2 highlighted in figure below around an axis coaxial to V.

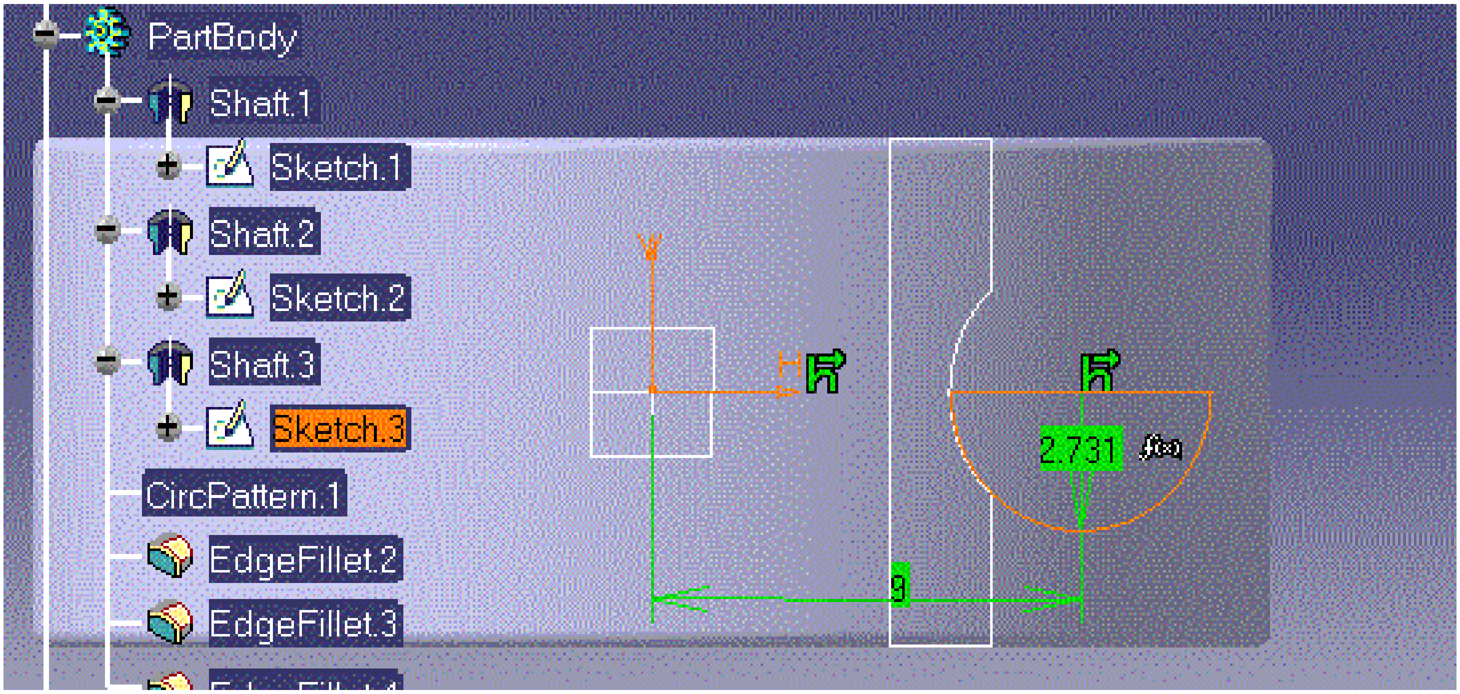


Here are the constraints defined on this sketch:

d2	2.1 mm	*ring width*
L2	3 mm	*half height of the non - hollowed inner surface*
B2	5 mm	*half height of the outer surface*
R2	2.759 mm	*groove radius*
b2	0 mm	*ordinate of the groove center*
D2	7.1 mm	*internal diameter*

The Balls

A ball is a shaft created by rotating half a circle (sketch.3) around the H axis. The circle must be closed before being rotated.



The parameters of the circular pattern which is created to build the set of balls are constrained by the formulas below:

- BallNumber = int(3* D3 / BallRadius)
- AngStep = 3.6deg / BallNumber

D3 being the abscissa of the ball center.

The Import File

In the scenario, you have to import the text file below which is delivered under the KwrBallBearingImport.txt name.

```
Temperature	100Kdeg		Maximum temperature allowed
Pressure	190N_m2	Maximum pressure allowed
LubricantVolume 0mm3		L1*D3*B1*0.005	required lubricant volume
```

If you modify this file, pay attention to the column format, use the Tab key to skip from one column to the other.

The Excel Table which Controls the Bearing Design

You must download the KwrBearingDesignTable.xls Excel table in your environment.

The CATScript Macro

The KwrBearing.CATScript macro just creates a circular pad. You can record this macro on your own in the Part Design workbench or use the one supplied with the KnowledgeAdvisor samples.

When creating **Rule.2** in your own environment, you should replace the pathname given as the argument of the LaunchMacroFromFile function with the pathname corresponding to the file where the macro has been downloaded.

Step-by-Step Procedure

Controlling the Bearing Design with a Design Table

A design table is created from a pre-existing file. The data set contained in this pre-existing file is quite similar to the data set which identifies a bearing in a catalogue. The design table which is created defines a number of configurations. Applying a new configuration results in a bearing modification.

1. Open the KwrBallBearing1.CATPart document.
2. Click the Design Table icon in the standard toolbar.
3. Check the **Create a design table from a pre-existing file** option. Click **OK**.
4. Select the KwrBearingDesignTable.xls file and associate automatically the design table columns and the document parameters (i.e. click **YES** in the "Automatic Associations?" dialog box).
5. In the **Design table** dialog box, select the configuration 3 (Line 3) and click **Apply**. Your ball bearing has changed. It is now a bronze bearing with 21 balls. You can tell the difference when you look at the geometry area. The bearing width is also modified. Click **OK** to exit the Design Table dialog box.
6. Keep your document open and proceed to the next task.

Creating a Check

A combined check using the => syntax is created. This check is intended to display a message whenever the check is not satisfied.

1. Access the Knowledge Advisor workbench

2. Click the icon then click **OK** in the first **Check Editor** dialog box. The check editor is displayed.

3. In the **Check Editor**, select the Warning type and enter the string "BallNumber is too small" in the message field.

   Then enter the

   D3 >= 6mm => BallNumber > 6

   relation in the edition box.

4. Click OK to create your check and exit the editor. At this stage, no particular message is displayed. The check is added to the specification tree with a green icon. For the configuration 3 of the design table, this is the status of the check relations:

   OK => OK

5. In the specification tree, double-click the design table and select the configuration 1. Click **OK**. The message "BallNumber is too small" is displayed. For the configuration 1 of the design table, this is the status of the check relations:

   OK => KO

Keep your document open and proceed to the next task

Creating a Multiple Value Parameter

A multiple value parameter is created. Depending on this parameter value, a rule which is created in the next task will display either a message or launch a macro.

1. Click the icon.
2. In the **Formulas** dialog box, select String in the **New Parameter of type** list. Select **Multiple values** in the with list, then click '**New Parameter of type**'.
3. In the Value List of String dialog box, enter one-by-one the step1, step2 and step3 values. Click OK.
4. In **Edit name or value of the current parameter**, replace the String.1 string with Status, then click **OK**. The Status parameter is added to the specification tree.

Creating a Rule

This task creates a rule which displays a message prompting you to import a file or launches a macro.

1. In the specification tree, double-click the design table feature and select the configuration 3 in the table which is displayed. You are back to 21 ball bearing.
2. Access the Knowledge Advisor workbench
3. Click the icon.
4. Enter the **Rule.2** string in the Name field of the first dialog box. Click **OK**.
5. Copy/Paste the code below into the rule edition box (modify the macro path):

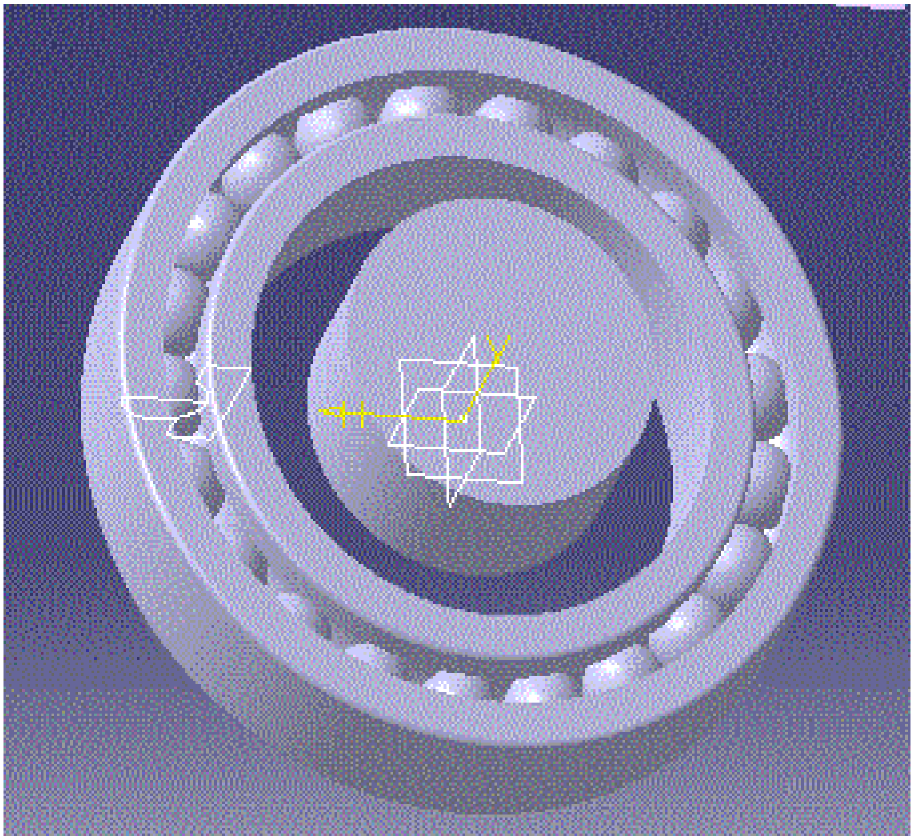
```
if Status == "step2"
Message("Import the KwrBallBearingImport text file")
else if Status == "step3"
LaunchMacroFromFile("e:/tmp/KwrBearing.CATScript")
```

6. Click **OK** to add the rule to the document and execute it.
7. Click the icon. In the "Formulas" dialog box, select the Status parameter and replace its step1 value with step2. Click **OK**. A message asks you to import the

KwrBallBearingImport text file.

8. Click **Import** and select the KwrBallBearingImport.txt file. Three parameters are then added to the document. Click OK in the dialog box displaying the parameters and formulas to be imported.

9. Select the Status parameter and replace the step2 value with step3. Click **OK**. The KwrBearing.CATScript is executed and a circular pad is created.



System of Three Equations in Three Variables

When designing a product, you may come across a system of equations to be solved. Whatever these equations (linear or not), *CATIA* provides you with resolution methods. These methods are the ***Simulated Annealing algorithm*** and the ***"SetOfEquations"*** capability.

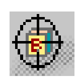
Can you use either method ?
If your set of equations is purely mathematical, the answer is yes. Otherwise, no. The SetOfEquations capability cannot solve systems using *CATIA* functions such as measures.
To solve a system of equations using measures, you must use the Simulated Annealing algorithm.
The Simulated Annealing algorithm is provided with the Product Engineering Optimizer product. The set of equations is to be specified as constraints and the variables are to be specified as free parameters. This resolution method is quite good although sometimes a bit long and you can use it to solve a broad range of cases. The trick about this algorithm is to adjust the precision and the other algorithm parameters. The example developed below works well with both methods. Just to illustrate a system that cannot be solved by both methods, you can draw a cube and create two user parameters: CubeSurface (of Area type) and CubeVolume (of Volume type). To calculate CubeSurface and CubeVolume, you can write either:

CubeSurface = smartWetarea (PartBody\Pad.1)
CubeVolume = smartVolume (PartBody\Pad.1)

or

CubeVolume = smartVolume (PartBody\Pad.1)

Solving the System of Equations by a Simulated Annealing

1. Open a new part document.
2. Create six real type parameters by using the f(x) capabilities. Name these parameters x1, y1, z1 and x2, y2, z2.
3. Access the Product Engineering Optimizer product and click the  icon.
4. In the Constraints tab, specify the three constraints (enter the constraints one-by-one)

   x1 + y1 - z1 == 0

   x1*y1 - z1 == 0

   sin(x1*1rad)2 - y1 - 1 == 0**

   Specify a precision of 0.01 for all three constraints.

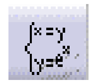
   If need be, see the *Product Engineering Optimizer User's Guide*.
5. In the Problem tab, specify x1, y1, z1 as free parameters and 1 as Step value for all three parameters.
6. Run the optimization process in Simulated Annealing mode. You can use the default termination criteria.

After the process has finished running, the x1, y1 and z1 values are close to the one below:

x1 = 0.454
y1 = -0.807
z1 = -0.363

Keep your document open and proceed to the next task.

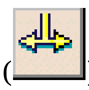
Solving the System of Equations by the "SetOfEquations" Capability

1. Access the Knowledge Advisor workbench, then click the  icon.
2. In the "Set of Equations" editor, enter the set of equations below:

   x2 + y2 == z2 ;

   x2*y2 == z2;

   sin(x2*1rad)2 == y2 +1**

   Specify x2, y2 and z2 as Unknown parameters by using the Parse arrow button ().
3. Click OK. The system of equations is solved. The values below are displayed in the specification tree

   x2 = 0.448043478

   y2 = -0.812335288

   z2 = -0.364229828

Knowledge Advisor Interoperability

This topic shows how to create Knowledge Advisor relations using technological packages and how to store them in ENOVIA VPM V5.

To know more about technological packages, see the CATIA Infrastructure User's Guide.

Optimal CATIA PLM Usability for Knowledge Advisor

When working with ENOVIA VPM V5, the safe save mode ensures that you only create data in *CATIA* that can be correctly saved in ENOVIA VPM V5.

ENOVIA VPM V5 offers two different storage modes: Workpackage (Document kept - Publications Exposed) and Explode (Document not kept).

Knowledge Advisor Commands in Enovia V5

Please find below the list of the Knowledge Advisor commands along with their accessibility status in ENOVIA VPM V5.

Please find below the supported Knowledge Advisor features:

Commands	**Accessibility in Enovia V5 (Explode Mode)**	**Comments**
Rule	Available	None
Check	Available	None
Reactions	Available	None
List	Available	None
Loop	Available	None
Add Set of Parameters	Available	None
Add Set of Relations	Available	None
Parameters Explorer	Available	None
Comment & URLs	Available	None
VB Macros with arguments	Available	None
Actions	Available	None
Measures update	Available	None
Set of Equations	Available	None

Saving a Product Structure Containing a Rule in ENOVIA VPM V5

This task explains how to save a product containing a rule in ENOVIA VPM V5. In this scenario, the user:

- Creates a product in ENOVIA VPM V5 and sends it to CATIA.
- Adds 2 components to the product as well as a rule that drives the 2 added components. He then stores the product in ENOVIA VPM V5.

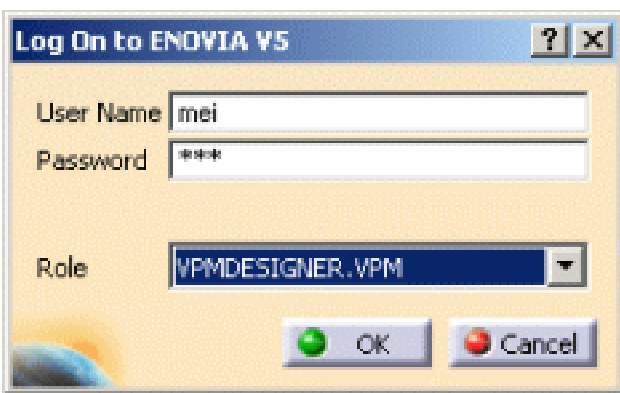
- Make sure you have selected **ENOVIAV5** in the **View -> Toolbars** menu.

1. Create a Product in ENOVIA VPM V5. (In this scenario the product is named Advisor).

   Note that you can close ENOVIA VPM V5 after creating the Product.

Sending the product to CATIA

2. To send the product to CATIA, proceed as follows:

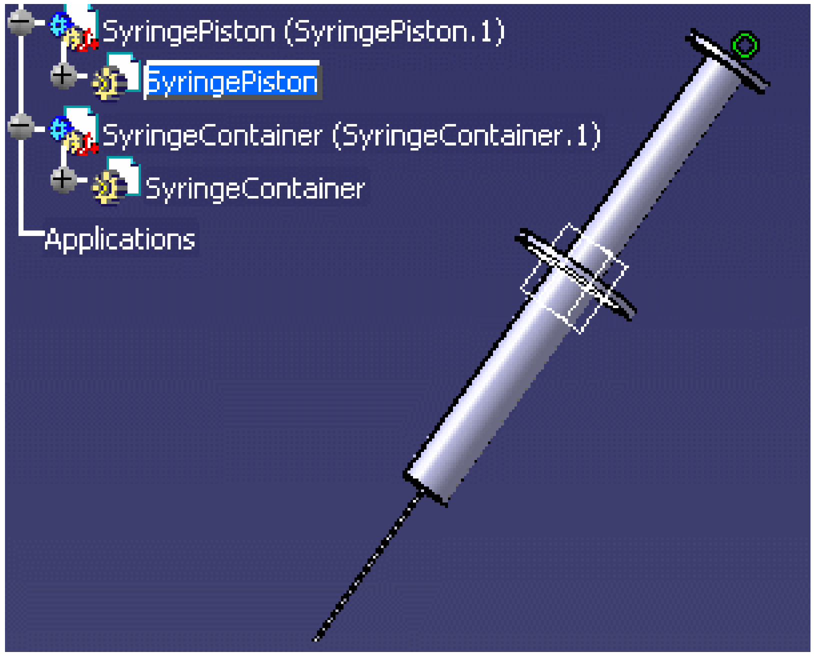


   - In CATIA, click the **Connect** icon () in the **ENOVIA V5** toolbar. The Log On to dialog box displays.
   - Enter your User Name, and your password.
   - Select your role in the Role scrolling list and click **OK**.
   - Click the **Search ENOVIA** icon (). The VPM Search dialog box displays.
   - In the Objects scrolling list, select the **Product** option.
   - Enter the Owner's ID.
   - Click **OK** when done. The Product ID displays in the Result window.
   - Double-click the product. The VPM Navigator view displays.
   - Right-click the root product (Advisor) and select the **Open** command. The Open Modes dialog box displays. Leave the default settings and click **OK**. The CATIA view displays.

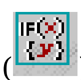
Adding components to the product

3. Click the root product and select the **Insert->Existing Component...** command. Select the KwrSyringeContainer.CATPart document. The syringe container is inserted into the product.

4. Click the root product and select the **Insert->Existing Component...** command. Select the KwrSyringePiston.CATPart document. The syringe container is inserted into the product.



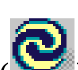
Adding a rule to the product

5. From the **Start->Knowledgeware** menu, access the **Knowledge Advisor** workbench and click the **Rule** icon () to create a rule.

6. Enter the rule body below using the parameters available in the Members of All column in the Rule Editor, and click **OK** when done.

```
SyringeContainer\PartBody\Sketch.2\Radius.2\Radius =
SyringePiston\PartBody\Sketch.2\Radius.2\Radius
+5mm
SyringeContainer\PartBody\Sketch.1\Radius.1\Radius =
SyringePiston\PartBody\Sketch.1\Radius.1\Radius
```

The rule is applied to the document.

7. From the **Start->Mechanical Design** menu, access the **Assembly Design** workbench and click the **Update All** icon () to update the document.

8. Right-click the technological package, and select the **Properties** command. Rename the

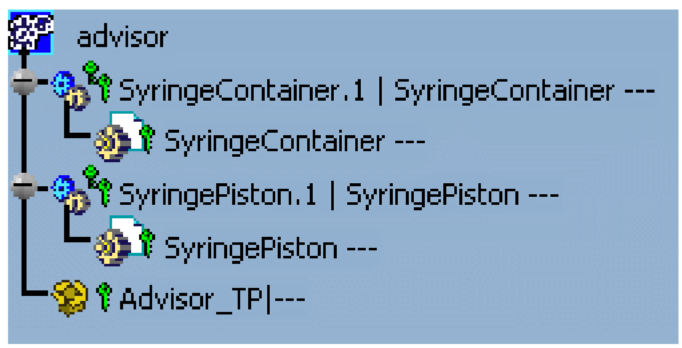
technological package: Advisor_TP in this scenario.

Saving Data in ENOVIA VPM V5

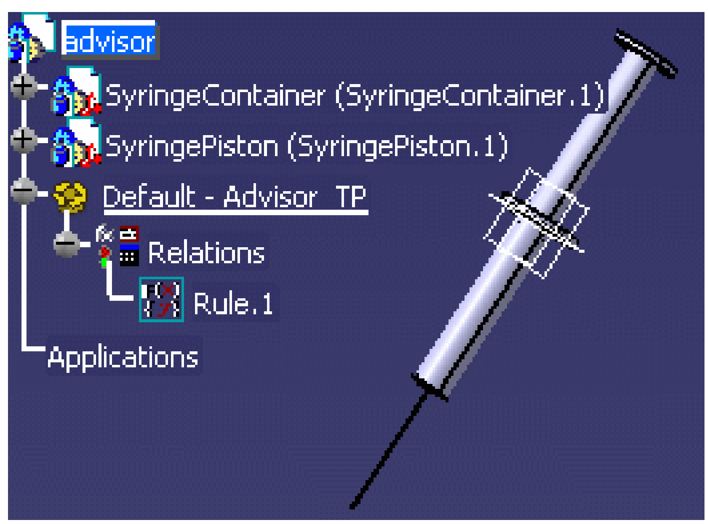
9. Click the **Save data in Enovia V5** icon (). The Save in ENOVIA dialog box displays. Click **OK**. Your document is saved in ENOVIA. Close the product in CATIA.

10. Send again your product to ENOVIA. To do so, proceed as follows:

   - Close all windows in CATIA.
   - Click the **Search ENOVIA** icon (). The VPM Search dialog box displays.
   - In the Objects scrolling list, select the **Product** option.
   - Enter the Owner's ID.
   - Click **OK** when done. The Product ID displays in the Result window.
   - Double-click the product. The VPM Navigator view displays.



   - Right-click the root product (Advisor) as well as the Parts and the technological package and select the **Open...** command. The Open Modes dialog box displays. Leave the default settings and click **OK**. The CATIA view displays.



Reference

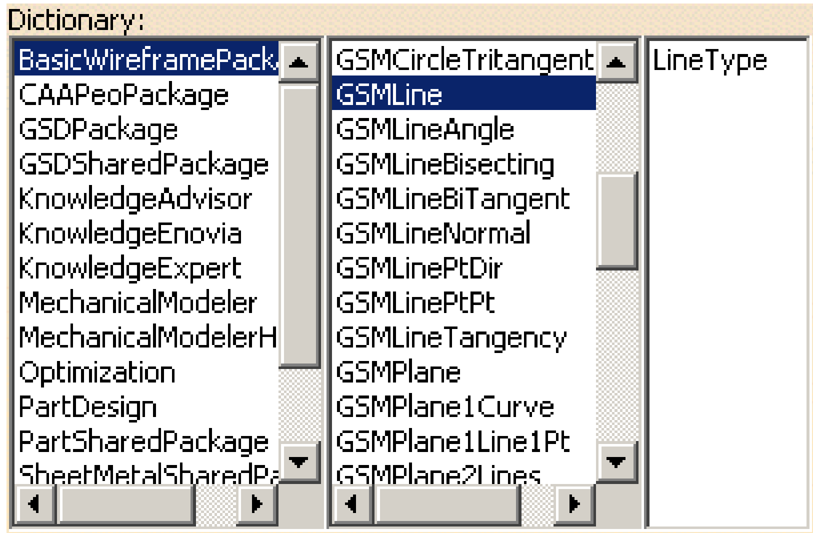
The packages listed below are those displayed in the Browser when specifying a loop body.

Basic Wireframe Package	GSD Package
GSD Shared Package	Knowledge Expert
Mechanical Modeler	Part Design
Part Shared Package	Standard

Basic Wireframe Package

GSMLine

Definition:



A GSMLine is a line :

- generated by the Generative Shape Design product.
- available in the BasicWireFrame Package.

To know more about lines, see the Generative Shape Design User's Guide.

Attributes:

LineType

A line is defined by its type. The attribute to be used is *LineType*. The syntax to be used is: **LineType = i**, i corresponding to the type of line that you want to create.

Please find below an equivalence table listing the existing types of lines that you can create and the digit to indicate.

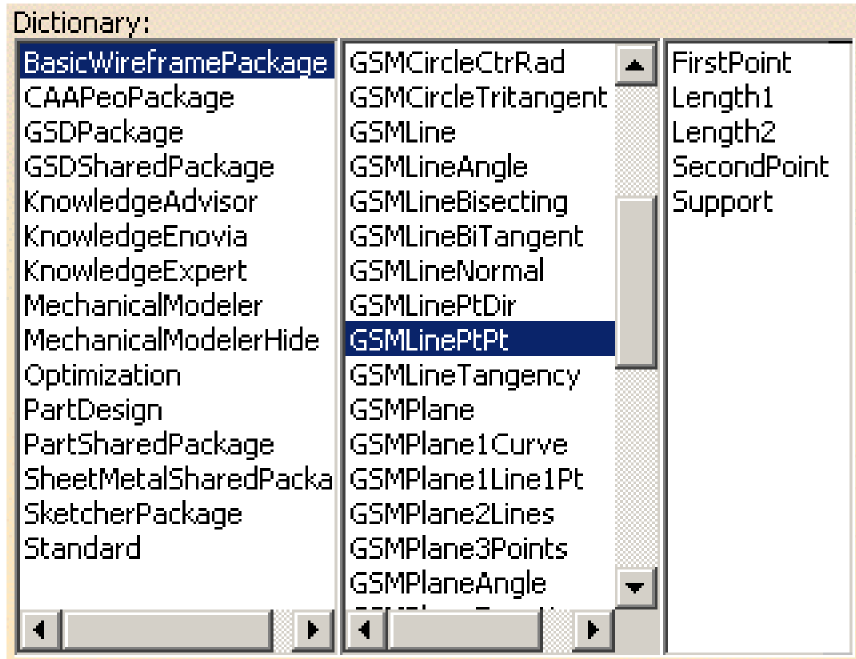
Line Type in GSD	Line Type in the Package	Corresponding digit
Point to Point	GSMLinePtPt	0
Point-Direction	GSMLinePtDir	1
Angle to Curve	GSMLineAngle	2
Tangent to Curve	GSMLineTangency	3
Normal to surface	GSMLineNormal	4
Intersection betw. 2 planes	GSMLineBiTangent	5

As mentioned above, you may create 7 different line sub-types. Please find below a description of each sub-type, as well as its attributes and the syntax to use.

Point to Point Line (*GSMLinePtpt*)

The sub-type to be used in this case is *GSMLinePtpt* which defines the line extremities. The following attributes are available for this sub-type:

- FirstPoint (feature)
- SecondPoint (feature)
- Support (feature)
- Length1 (length, optional for both combinations)
- Length2 (length, optional for both combinations)



These attributes can be combined as follows:

1st combination	2nd combination
• the *FirstPoint* which is defined by the syntax below: FirstPoint = object: ..\..*theFirstPoint*; • the *SecondPoint* which is defined by the syntax below: SecondPoint = object: ..\..*theSecondPoint*; • Length1 which is defined by the syntax below: Length1=200mm; • Length2 which is defined by the syntax below: Length2=150mm;	• the *FirstPoint* which is defined by the syntax below: FirstPoint = object: ..\..*theFirstPoint*; • the *SecondPoint* which is defined by the syntax below: SecondPoint = object: ..\..*theSecondPoint*; • the Support

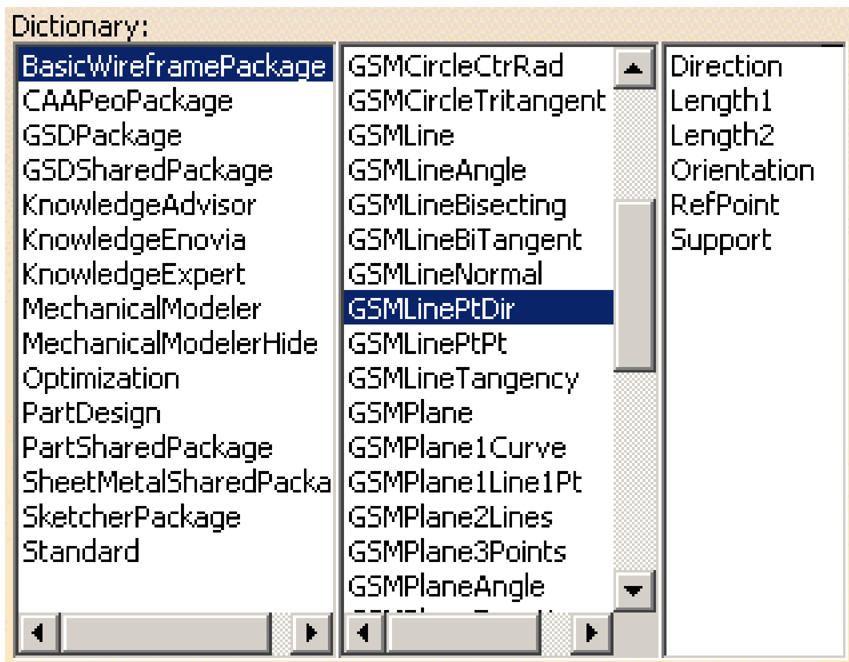
```
Line.1 isa GSMLine
{
    LineType = 0;
    TypeObject isa GSMLinePtPt
    {
      FirstPoint = object: Point.1;
      SecondPoint = object: Point.2;
      Length1 = 500mm; \\ optional
      Length2 = 435mm; \\ optional
    }
  }
```

```
Line.2 isa GSMLine
  {
   LineType = 0;
   TypeObject isa GSMLinePtPt
   {
     FirstPoint = object: ..\..\..\Point.1;
     SecondPoint = object: ..\..\..\Point.2;
     Support = object: ..\..\..\Extrude.1;
     Length1 = 50mm; \\ optional
     Length2 = 45mm; \\ optional
   }
  }
```

Point-Direction (*GSMLinePtDir*)

The sub-type to be used in this case is *GSMLinePtDir* which defines the line direction. The following attributes are available for this sub-type:

- Length1
- Length2
- Direction
- Orientation
- RefPoint
- Support



These attributes can be combined as follows:

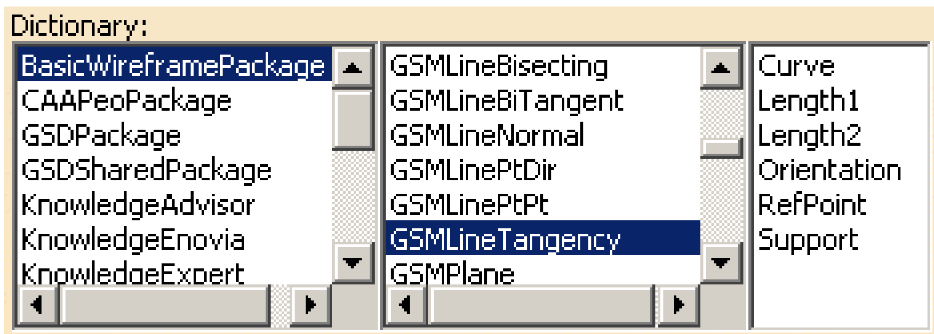
Combination

- Length1 which is defined by the syntax below:
  Length1 = 100mm;
- Length2 which is defined by the syntax below:
  Length2 = 10mm;
- Direction which is defined by the syntax below:
  Direction = object: ..\..\Plane.2;
- *RefPoint* which is defined by the syntax below:
  RefPoint = object: ..\..\Point.2;
- Support which is defined by the syntax below:
  SecondPoint = object: ..\..*'xy plane'*;

Tangent to Curve (*GSMLineTangency*)

The sub-type to be used in this case is *GSMLineTangency*. The following attributes are available for this sub-type:

- Curve: Reference curve used to define the tangency.
- Length1
- Length2
- Orientation
- RefPoint: Reference point used to define the tangency.
- Support



These attributes can be combined as follows:

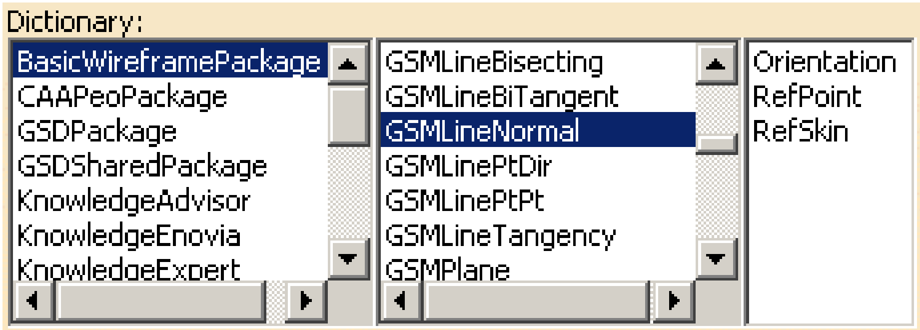
Combination

- Curve which is defined by the syntax below:
  Curve = object: ..\..\Spline.2;
- Length1 which is defined by the syntax below:
  Length1 = 100mm;
- Length2 which is defined by the syntax below:
  Length2 = 10mm;
- *RefPoint* which is defined by the syntax below:
  RefPoint = object: ..\..\Point.2;
- Support which is defined by the syntax below:
  SecondPoint = object: ..\..*'xy plane'*;

Normal to surface (*GSMLineNormal*)

The sub-type to be used in this case is *GSMLineNormal*. The following attributes are available for this sub-type:

- Orientation
- RefPoint
- RefSkin



These attributes can be combined as follows:

Combination

- *RefPoint* which is defined by the syntax below:
  RefPoint = object: ..\..\Point.2;
- Support which is defined by the syntax below:
  RefSkin = object: ..\..\Extrude.1;

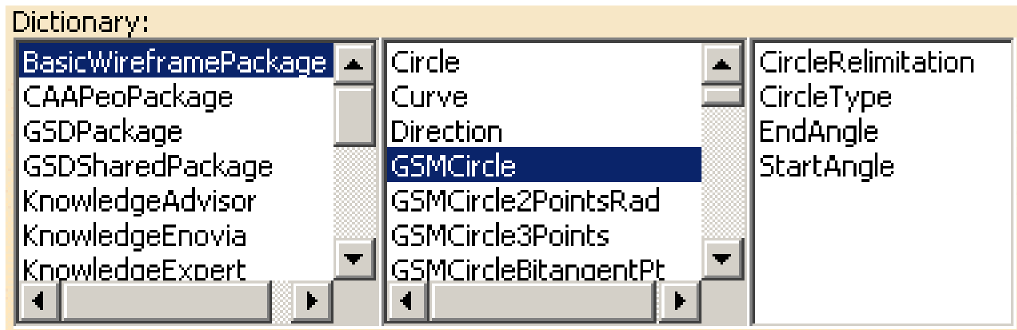
GSMCircle

Definition:

A GSMCircle is a circle:

- generated by the Generative Shape Design product.
- available in the BasicWireFrame Package.

To know more about circles, see the Generative Shape Design User's Guide.



Attributes:

PointType

A point is defined by the following attributes:

- *CircleType:* The syntax to be used is **CircleType = i**, i corresponding to the type of circle that you want to create.
- *CircleRelimitation:* The syntax to be used is **CircleRelimitation =**.
- *EndAngle:* The syntax to be used is **EndAngle =** xxxdeg.
- *StartAngle:* The syntax to be used is **StartAngle =**xxxdeg.

Please find below a table listing the existing types of circles that you can create and the digit to indicate.

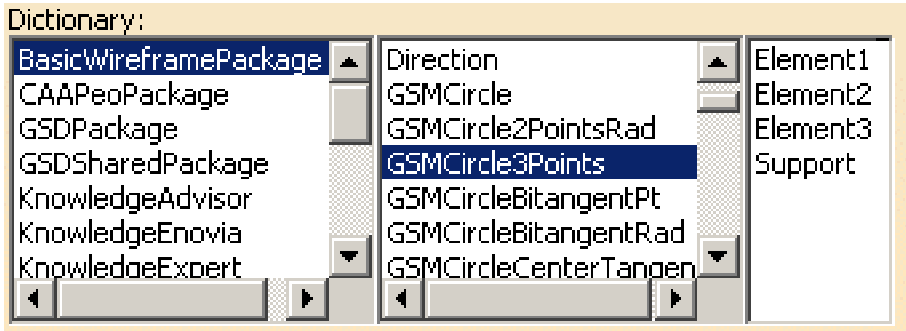
Plane Type in GSD	Plane Type in the Package	Corresponding digit
Three Points	GSMPCircle3Points	3
Center and Radius	GSMCircleCtrRad	0
Center and Point	GSMCircleCtrPt	1

As mentionned above, you may create 3 different circle sub-types. Please find below a description of each sub-type, as well as its attributes and the syntax to use.

Three Points (*GSMCircle3Points*)

The sub-type to be used in this case is *GSMCircle3Points* which enables you to create a circle passing through 3 points. The following attributes are available for this sub-type:

- Element1: First point
- Element2: Second point
- Element3: Third point
- Support: Support surface onto which the circle will be projected (optional)



These attributes can be combined as follows:

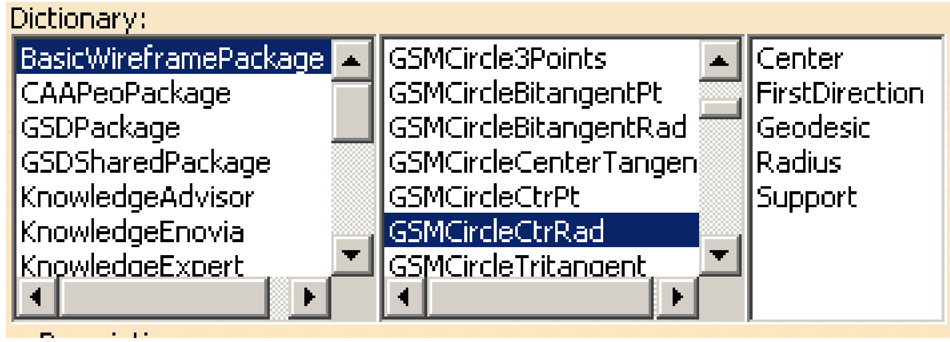
Combination

- Element1 which is defined by the syntax below:
  Element1 = object: ..\Point.1;
- Element2 which is defined by the syntax below:
  Element2 = object: ..\Point.2;
- Element3 which is defined by the syntax below:
  Element3 = object: ..\Point.3;
- Support which is defined by the syntax below:
  Support = object: ..\Extrude.1;

Center and Radius (*GSMCircleCtrRad*)

The sub-type to be used in this case is *GSMCircleCtrRad* which enables you to create a circle by indicating its center and its radius. The following attributes are available for this sub-type:

- Center: Point that will be the center of the circle.
- FirstDirection
- Geodesic
- Radius: Radius of the circle.
- Support: Support plane or surface onto which the circle is to be created.



These attributes can be combined as follows:

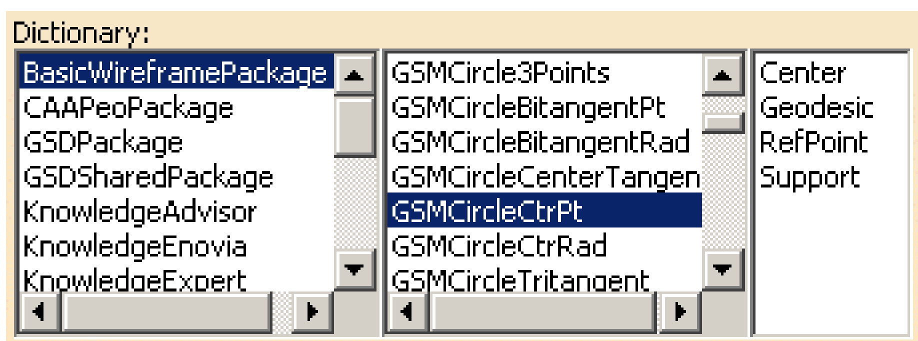
Combination

- Center which is defined by the syntax below:
  Center = object: ..\Point.1;
- Radius which is defined by the syntax below:
  Radius = 120mm;
- Support which is defined by the syntax below:
  Support = object: ..\Extrude.1;

Center and point (*GSMCircleCtrPt*)

The sub-type to be used in this case is *GSMCircleCtrPt* which enables you to create a circle by indicating its center and a point. The following attributes are available for this sub-type:

- Center: Point used as the center of the circle.
- Geodesic: Curve.
- RefPoint: Second point used to create the circle.
- Support: Support plane or surface where the circle is to be created.

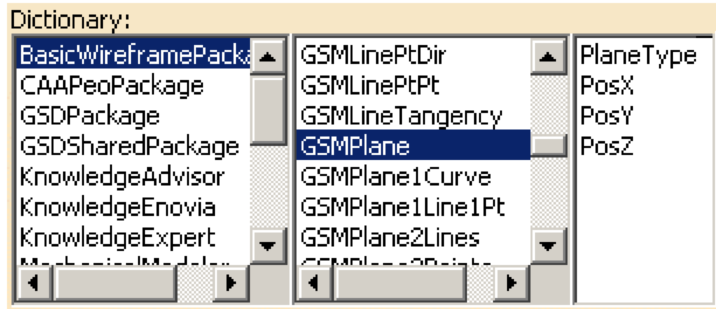


Combination

- Center which is defined by the syntax below:
  Center = object: ..\Point.1;
- RefPoint which is defined by the syntax below:
  RefPoint = object: ..\Point.1;
- Support which is defined by the syntax below:
  Support = object: ..\Extrude.1;

GSMPlane

Definition:



A GSMPlane is a plane:

- generated by the Generative Shape Design product.
- available in the BasicWireFrame Package.

To know more about planes, see the Generative Shape Design User's Guide.

Attributes:

PlaneType

A plane is defined by its type. The attribute to use is *PlaneType*. The syntax to be used is: **PlaneType = i**, i corresponding to the type of plane that you want to create.

Please find below a table listing the existing types of planes that you can create and the digit to indicate.

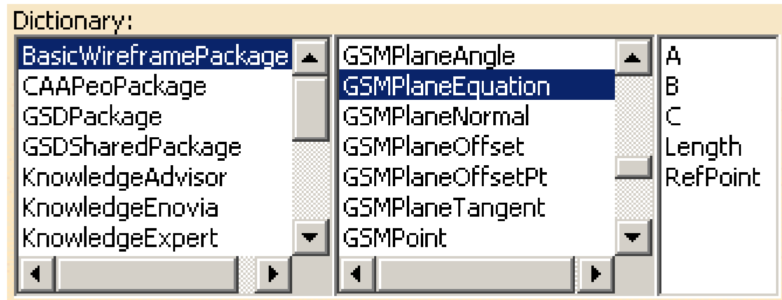
Plane Type in GSD	Plane Type in the Package	Corresponding digit
Equation	GSMPlaneEquation	0
Through 3 points	GSMPlane3Points	1
Through 2 lines	GSMPlane2Lines	2
Through a point and a line	GSMPlane1line1Pt	3
Normal to a curve	GSMPlane1Curve	4
Tangent to a surface	GSMPlaneTangent	5
Normal to a plane	GSMPlaneNormal	6

As mentionned above, you may create 7 different plane sub-types. Please find below a description of each sub-type, as well as its attributes and the syntax to use.

Equation (*GSMPlaneEquation*)

The sub-type to be used in this case is *GSMPlaneEquation* which enables you to create a plane by using an equation. The following attributes are available for this sub-type:

- A (First component of the equation)
- B (Second component of the equation)
- C (Third component of the equation)
- Length
- RefPoint (point used to position the plane through this point)



These attributes can be combined as follows:

1st Combination	2nd Combination

• A which is defined by the syntax below:
A=31; //A value is required
 	
• B which is defined by the syntax below:
B=-47; //A value is required
 	
• C which is defined by the syntax below:
C=-24; //A value is required
 	
• Length: enables the user to indicate the required length. It is defined by the syntax below:
Length=24mm
	• A which is defined by the syntax below:
	A=31; //A value is required
	
	• B which is defined by the syntax below:
	B=-47; //A value is required
	
	• C which is defined by the syntax below:
	C=-24; //A value is required
	
	• RefPoint which is defined by the syntax below:
	RefPOint = object: ..\Point ;

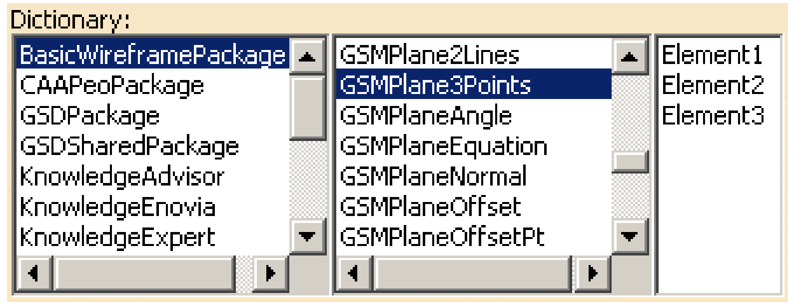
```
Plane0.1 isa GSMPlane
{
   PlaneType = 0;
   TypeObject isa GSMPlaneEquation
    {
      A = 15;
      B = -12;
      C = 31;
      Length = 24mm;
    }
}
```

```
Plane0.2 isa GSMPlane
{
   PlaneType = 0;
   TypeObject isa GSMPlaneEquation
    {
      A = 31;
      B = -47;
      C = -24;
      RefPoint = object: ..\ConstrBody\Point.5;
    }
}
```

Through 3 points (*GSMPlane3Points*)

The sub-type to be used in this case is *GSMPlane3Points* which creates a plane passing through 3 points. The following attributes are available for this sub-type:

- Element1 (First point)
- Element2 (Second point)
- Element3 (Third point)



These attributes can be combined as follows:

Combination

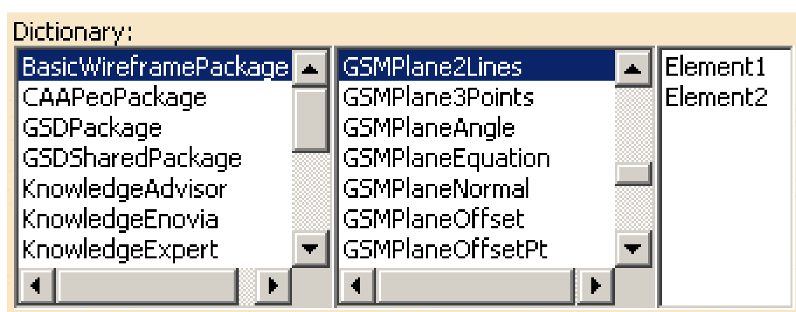
- Element1 which is defined by the syntax below:
  Element1 = object: ..\Point.1;
- Element2 which is defined by the syntax below:
  Element2 = object: ..\Point.2;
- Element3 which is defined by the syntax below:
  Element3 = object: ..\Point.3;

```
Plane1 isa GSMPlane
{
   PlaneType = 1;
   TypeObject isa GSMPlane3Points
            {
      Element1 = object: ..\Point.1;
      Element2 = object: ..\Point.5;
      Element3 = object: ..\Point.8;
            }
}
```

Through 2 Lines (*GSMPlane2Lines*)

The sub-type to be used in this case is *GSMPlane2Lines* which enables to create a plane passing through 2 lines. The following attributes are available for this sub-type:

- Element1 (First line)
- Element2 (Second line)



Combination

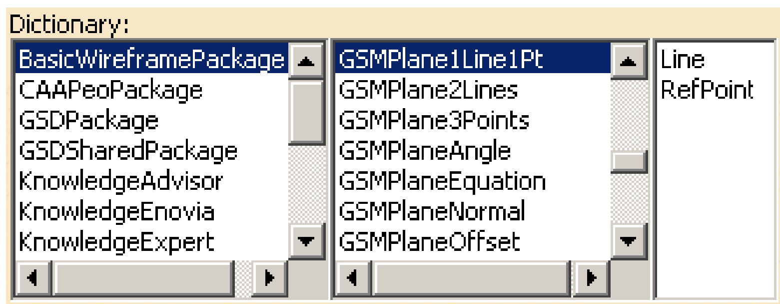
- Element1 which is defined by the syntax below:
  Element1 = object: ..\Line.1;
- Element2 which is defined by the syntax below:
  Element2 = object: ..\Line.2;

```
Plane2 isa GSMPlane
{
   PlaneType = 2;
   TypeObject isa GSMPlane2Lines
            {
                Element1 = object: ..\Line.1;
                Element2 = object: ..\Line.2;
            }
}
```

Through a Point and a Line (*GSMPlane1line1Pt*)

The sub-type to be used in this case is *GSMPlane1Line1Pt* which enables to create a plane passing through a line and a point. The following attributes are available for this sub-type:

- Line: Line used to create the plane.
- RefPoint: Point used to create the plane.



The attributes should be used as follows:

Combination

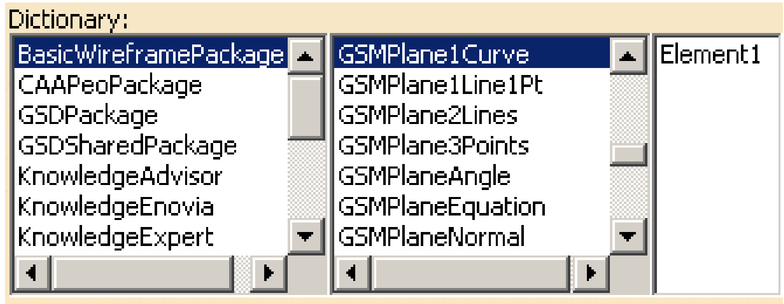
- Line which is defined by the syntax below:
  Line = object: ..\Line.1;
- RefPoint which is defined by the syntax below:
  RefPoint = object: ..\Point.2;

```
Plane3 isa GSMPlane
{
   PlaneType = 3;
   TypeObject isa GSMPlane1line1Pt
            {
                Line = object: ..\Line.1;
                RefPoint = object: ..\Point.13;
            }
}
```

Normal to a Curve (*GSMPlane1Curve*)

The sub-type to be used in this case is *GSMPlane1Curve* which enables you to create a plane normal to a curve at a specified point.

- Element1: Line



This attribute is to be used as follows:

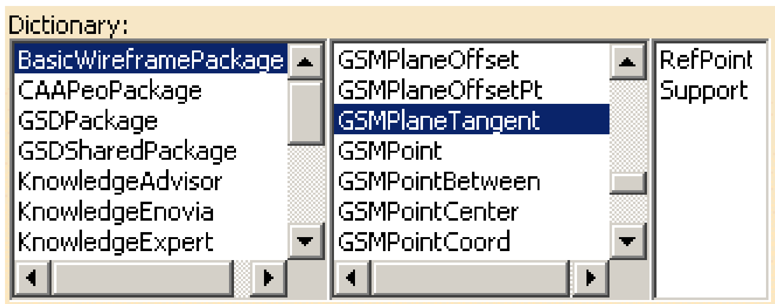
Combination

- Line which is defined by the syntax below:
  Line = object: ..\Spline.1;

Tangent to a Surface (*GSMPlaneTangent*)

The sub-type to be used in this case is *GSMPlaneTangent* which enables you to create a plane tangent to a surface at a specified point. The following attributes are available for this sub-type:

- RefPoint (Point)
- Support (Surface)



These attributes are to be used as follows:

Combination

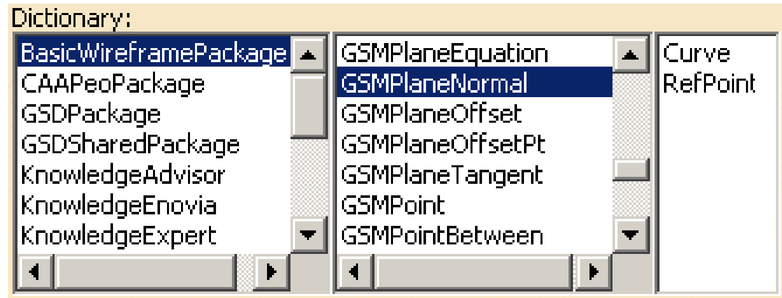
- Support which is defined by the syntax below:
  Support = object: ..\Spline.1;
- RefPoint which is defined by the syntax below:
  RefPoint = object: ..\Point.4;

```
Plane5 isa GSMPlane
{
   PlaneType = 5;
   TypeObject isa GSMPlaneTangent
            {
                Support = object: ..\Extrude.1;
                RefPoint = object: ..\Point.4;
            }
}
```

Normal to a Plane (GSMPlaneNormal)

The sub-type to be used in this case is *GSMPlaneNormal*. The following attributes are available for this sub-type:

- Curve: Reference curve used to create the plane.
- RefPoint: Reference point used to create the plane.



These attributes are to be used as follows:

Combination

- Curve which is defined by the syntax below:
  Curve = object: ..\Spline.1;
- RefPoint which is defined by the syntax below:
  RefPoint = object: ..\Point.4;

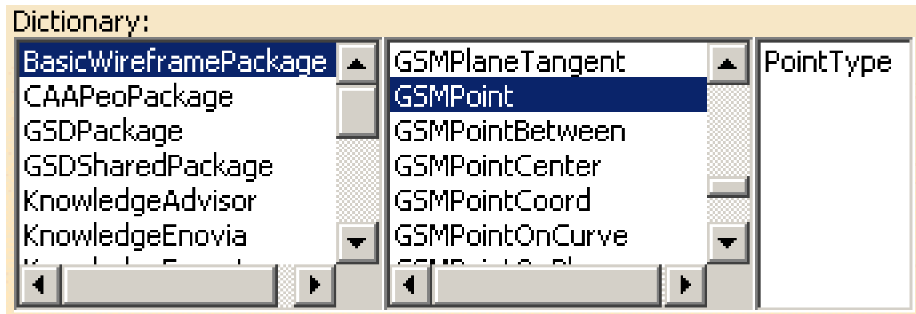
```
Plane6 isa GSMPlane
{
   PlaneType = 6;
   TypeObject isa GSMPlaneNormal
            {
                Curve = object: ..\Spline.2;
                RefPoint = object: ..\Point.13;
            }
}
```

GSMPoint

Definition:

A GSMPoint is a point:

- generated by the Generative Shape Design product
- available in the BasicWireFrame Package.



To know more about points, see the Generative Shape Design User's Guide.

Attributes:

PointType

A point is defined by its type. The attribute to use is *PointType*. The syntax to be used is: **PointType = i**, i corresponding to the type of point that you want to create.

Please find below a table listing the existing types of points that you can create and the digit to indicate.

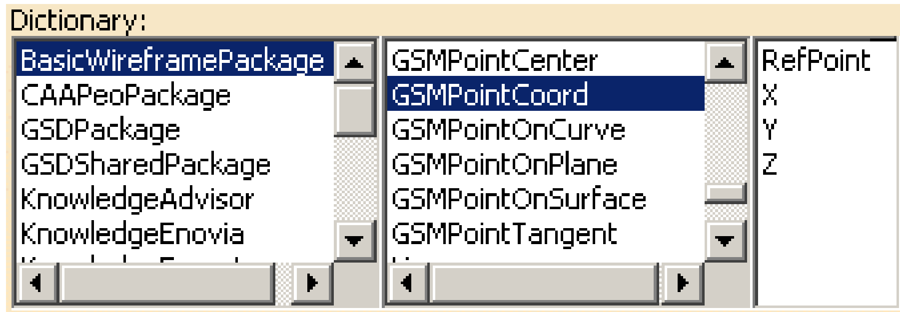
Plane Type in GSD	Plane Type in the Package	Corresponding digit
Coordinates	GSMPointCoord	0
On surface	GSMPointOnSurface	1
On curve	GSMPointOnCurve	2
On plane	GSMPointOnPlane	3
Circle center	GSMPointCenter	4

As mentionned above, you may create 5 different point sub-types. Please find below a description of each sub-type, as well as its attributes and the syntax to use.

Coordinates (*GSMPointCoord*)

The sub-type to be used in this case is *GSMPointCoord* which enables you to create a coordinate point. The following attributes are available for this sub-type:

- RefPoint (Reference point, optional). If specified, x, y, and z are indicated in a mark whose origin is this reference point.
- X (First coordinate)
- Y (Second coordinate)
- Z (Third coordinate)



These attributes can be combined as follows:

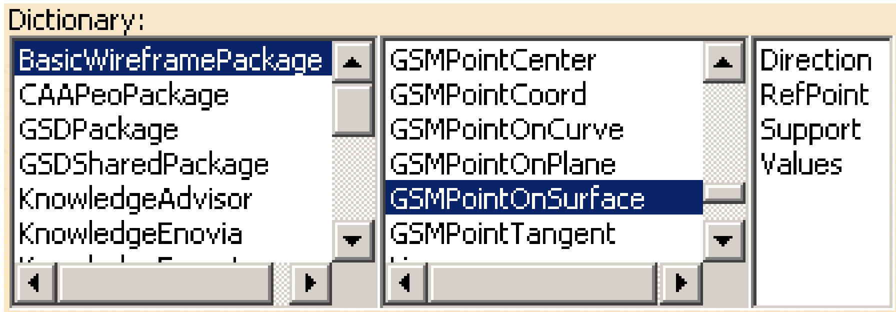
Combination

- RefPoint (Reference point, optional)
- X which is defined by the syntax below:
  X = 10mm;
- Y which is defined by the syntax below:
  Y = 10mm;
- Y which is defined by the syntax below:
  Z = 10mm;

```
Point0.2 isa GSMPoint
     {
       PointType = 0;
       TypeObject isa GSMPointCoord
       {
         RefPoint = object: ..\Construction_Body\GSMPoint.21;
         X= 12mm;
         Y= 15mm;
         Z= 18mm;
       }
     }
```

On surface (*GSMPointOnSurface*)

The sub-type to be used in this case is *GSMPointOnSurface* which creates a point on a plane. The following attributes are available for this sub-type:



- Direction: Element taking its orientation as reference direction or a plane taking its normal as reference direction
- RefPoint: Reference point. By default, the surface middle point is taken as reference.
- Support: Surface where the point is to be created.
- Values: Distance along the reference direction used to display a point.

These attributes can be combined as follows:

Combination

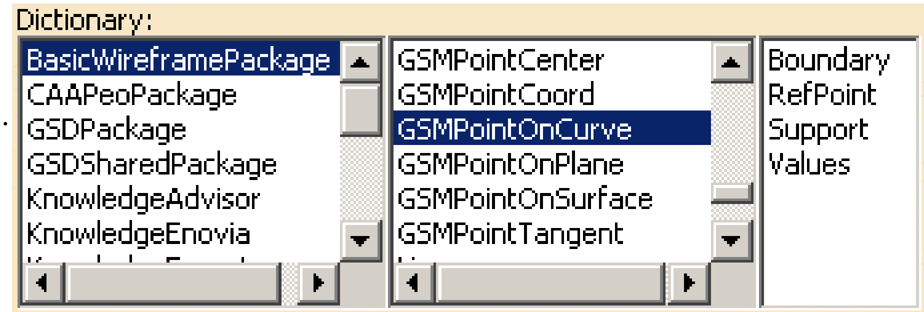
- Direction which is defined by the syntax below:
  Direction = object: ..\Line.1;
- Support which is defined by the syntax below:
  Support= object: ..\Extrude.1;
- Values which is defined by the syntax below:
  Values = 12mm;

```
Point1 isa GSMPoint
{
  PointType = 1;
  TypeObject isa GSMPointOnSurface
  {
  Direction = object: ..\Construction_Body\Line.4;
  Support = object: ..\Construction_Body\GSMExtrude.1;
  Values = 25mm;
  }
}
```

On curve (*GSMPointOnCurve*)

The sub-type to be used in this case is *GSMPointOnCurve* which enables to create a point on a curve. The following attributes are available for this sub-type:

- Boundary: Not available.
- RefPoint: Reference point. If not specified, it is the extremity of the curve.
- Support: Curve
- Values: Distance between the reference point and this point.



Combination

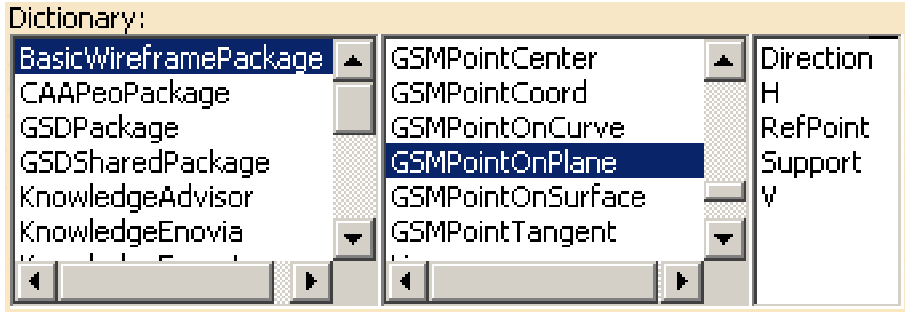
- Refpoint which is defined by the syntax below:
  RefPoint= object: ..\Point.1;
- Support which is defined by the syntax below:
  Support = object: ..\Line.1;
- Values which is defined by the syntax below:
  Values = 12mm;

```
Point2 isa GSMPoint
{
  PointType = 2;
  TypeObject isa GSMPointOnCurve
  {
  Support = object: ..\Construction_Body\GSMLine.3;
  Values = 125mm;
  }
}
```

On plane (*GSMPointOnPlane*)

The sub-type to be used in this case is *GSMPointOnPlane*. It creates a point on a plane. The following attributes are available for this sub-type:

- Direction (optional). When specified, indicates the direction
- H: Vector.
- RefPoint: point used to define a reference for computing coordinates in the plane.
- Support: Plane on which the point will be created.
- V: Vector.



The attributes should be used as follows:

Combination

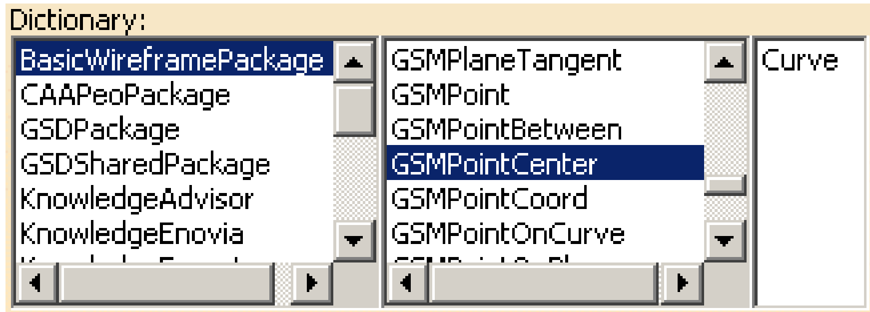
- Direction which is defined by the syntax below:
  Direction = object: ..\Line.1;
- H which is defined by the syntax below:
  H = 150mm;
- RefPoint which is defined by the syntax below:
  RefPoint= object: ..\Point.1;
- Support which is defined by the syntax below:
  Support = object: 'xy plane'
- V which is defined by the syntax below:
  V = 150mm;

```
Point3 isa GSMPoint
{
  PointType = 3;
  TypeObject isa GSMPointOnPlane
  {
  Support = object: ..\..\..\`xy plane`;
  H = 150mm;
  V = 120mm;
  }
}
```

Circle Center (GSMPointCenter)

The sub-type to be used in this case is *GSMPointCenter* which enables you to define the center of a circle.

- Curve: circle, circular arc, or ellipse.



This attribute is to be used as follows:

Combination

- Curve which is defined by the syntax below:
  Curve = object: ..\Extrude.1;

```
Point4 isa GSMPoint
{
  PointType = 4;
  TypeObject isa GSMPointCenter
  {
    Curve = object: ..\GSMExtrude.2;
           }
}
```

Part Design

Please find below a table listing the types available in the Part Design package.

Box	Chamfer	Cone
Counterbored Hole	Counterdrilled Hole	Countersunk Hole
Cylinder	Hole	Pad
Pocket	*RemoveFace*	*ReplaceFace*
Shaft	Shell	SimpleHole
SoldCombine	*Split*	TaperedHole
Thickness	ThickSurface	Torus

Box

Definition:

A box is a pad extruded from a rectangular sketch.

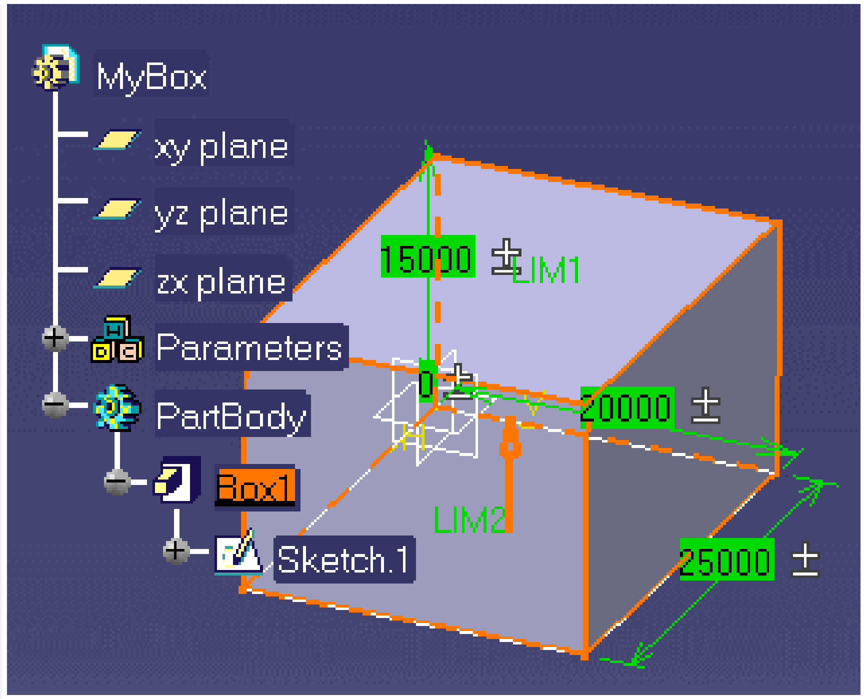
Attributes:

A box is defined by the following attributes:

- *Length* which is the pad first limit. The syntax to be used is **Length = 10mm**.
- *Width* which is the pad width. The syntax to be used is **Width = 20mm**.
- *Height* which is the pad height. The syntax to be used is **Height = 12mm**.

```
MyBox isa CATPart
 {
 BoxPart isa Part
  {
   PartBody isa BodyFeature
    {
    // Create a box
    Box1 isa Box
      {
       // Specify the box properties
       Width = 20.0 mm ;
       Height = 25.0 mm ;
       Length = 15.0 mm ;
      }
    }
  }
```

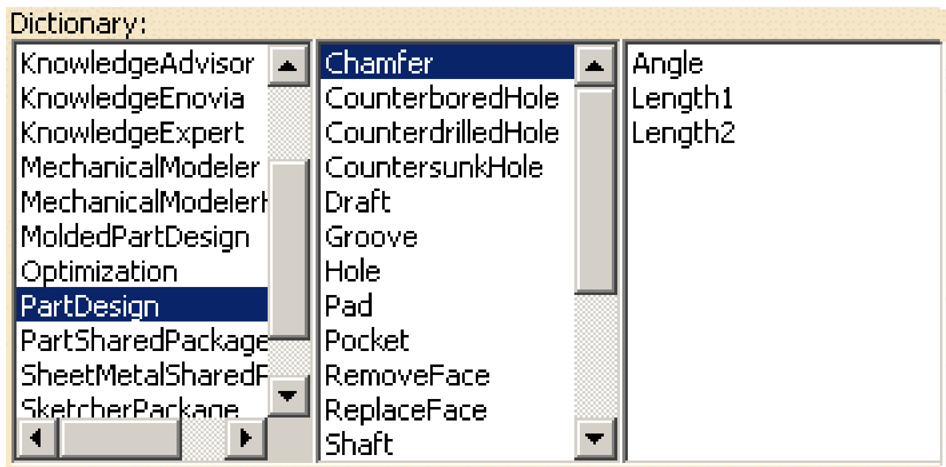
```
}
```



Chamfer

Definition:

A cut through the thickness of the feature at an angle, giving a sloping edge.



Attributes:

A chamfer is defined by the following attributes:

- *Angle*. The syntax to be used is **Angle = 20 deg;**
- *Length1*. The syntax to be used is **Length1 = 5 mm;**
- *Length2*. The syntax to be used is **Length2 = 5 mm;**

Important Notes:

- A chamfer has a Length2 attribute which is the default chamfer length. You don't have to manipulate this attribute in a script.

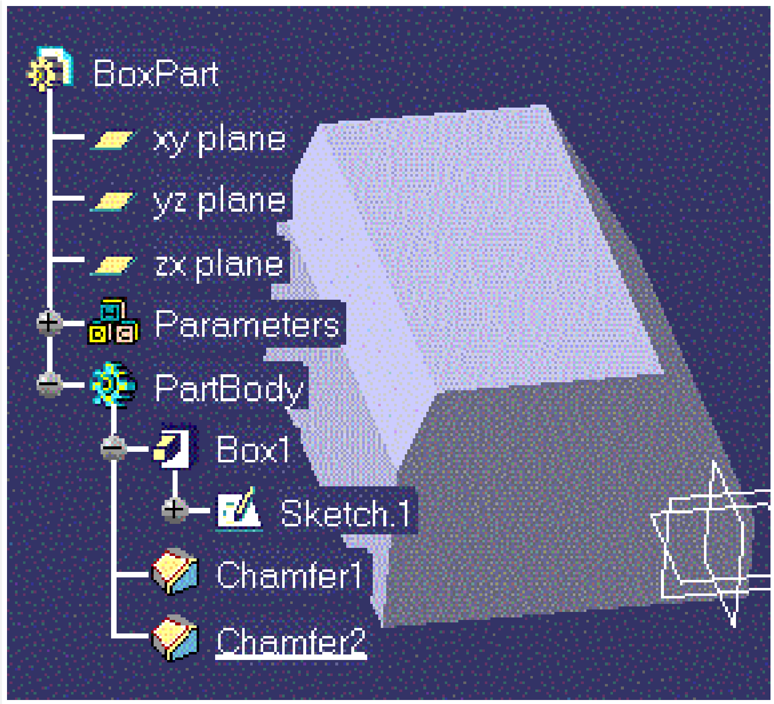
To specify a chamfer within your script, you must have a part open, then proceed as follows:

1. Create a Chamfer by using the isa function

   Chamfer1 isa Chamfer ( ) { }

2. Right-click anywhere inside the parentheses and select the 'Get Edge' or the 'Get Surface' command from the contextual menu. Then, in the geometry area, select the edge or surface to be chamfered.

```
MyBox isa CATPart
 {
   BoxPart isa Part
     {
       PartBody isa BodyFeature
         {
           // Create a box
           Box1 isa Box
             {
               Width = 20.0 mm ;
               Height = 25.0 mm ;
               Length = 15.0 mm ;
             }
           // Create a chamfer
           // The edge definition must be captured
           // from the geometry area
           // Use the Get Edge command from the
           // contextual menu
           Chamfer1 isa Chamfer (Edge Definition)
             {
               Angle = 20 deg;
               Length1 = 5 mm ;
             }
           Chamfer2 isa Chamfer (Edge Definition)
             {
               Angle = 30 deg;
               Length1 = 10 mm ;
             }
         }
     }
 }
```



Cone

Definition:

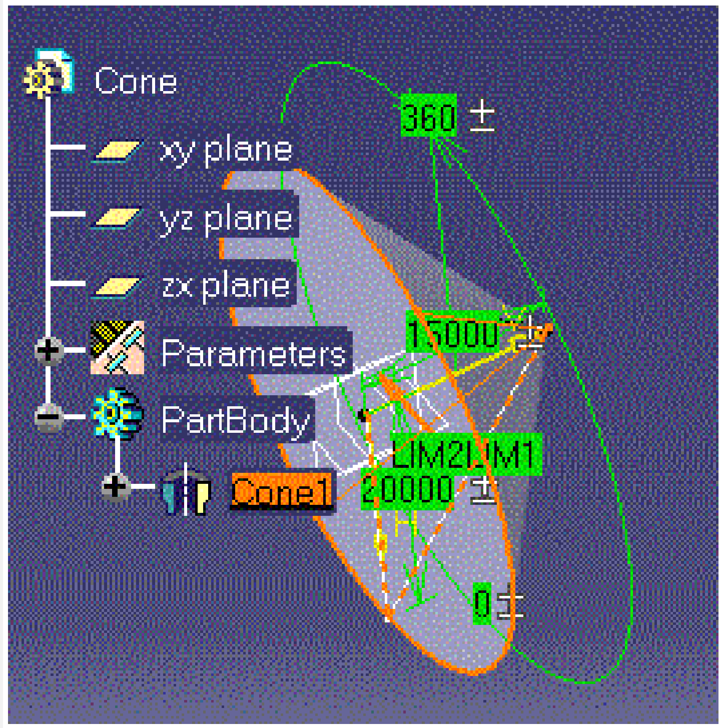
A cone is a shaft created by rotating a triangular sketch.

Attributes:

A cone is defined by the following attributes:

- Length. The syntax to be used is **Length = 15.0 mm ;**.
- Radius. The syntax to be used is **Radius = 20.0 mm ;**.

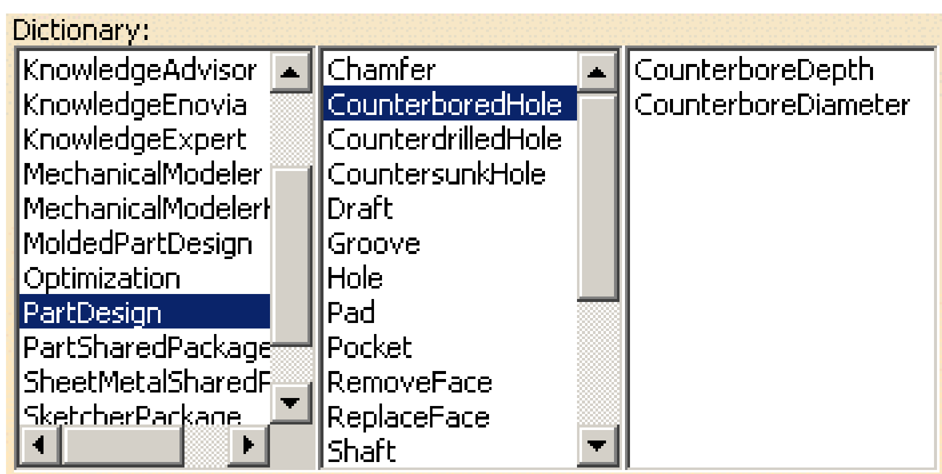
```
MyCone isa CATPart
  {
    ConePart isa Part
      {
        PartBody isa BodyFeature
          {
            // Create a cone
            Cone1 isa Cone
              {
                Radius = 20.0 mm ;
                Length = 15.0 mm ;
              }
          }
      }
  }
```

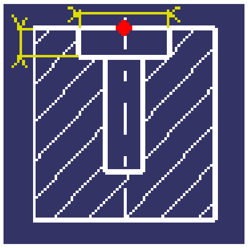


Counterbored Hole

Definition:

A mechanical feature of Hole type you create when you click the icon in the Part Design workbench. For more information, refer to the *Part Design User's Guide*.





Attributes:

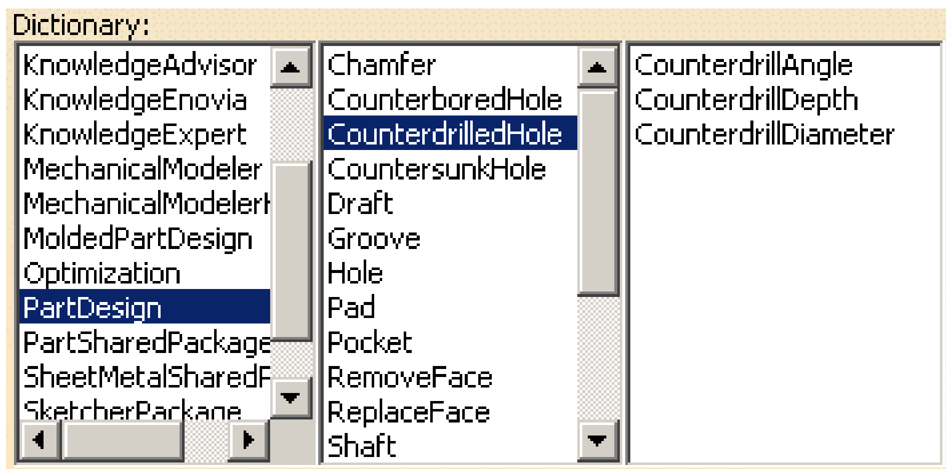
A counterbored hole is defined by the following attributes:

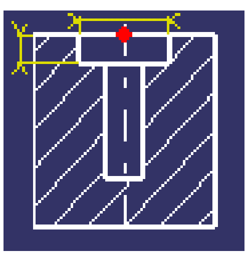
- CounterboreDepth. The syntax to be used is **CounterboreDepth =12mm**.
- CounterboreDiameter: The syntax to be used is **CounterboreDiameter =15mm**.

Counterdrilled Hole

Definition:

A mechanical feature of Hole type you create when you click the  icon in the Part Design workbench. For more information, refer to the *Part Design User's Guide*.





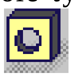
Attributes:

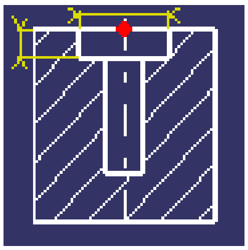
A counterdrilled hole is defined by the following attributes:

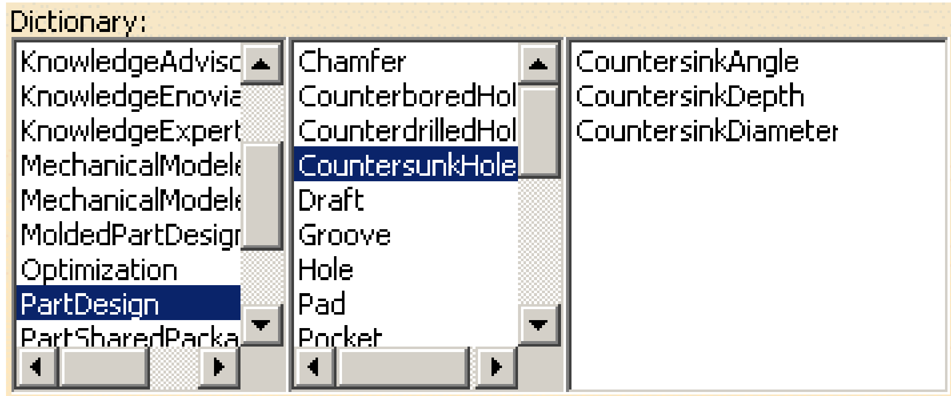
- CounterdrillAngle. The syntax to be used is **CounterdrillAngle = 22deg**.
- CounterdrillDiameter. The syntax to be used is **CounterdrillDiameter** =12mm.
- CounterdrillDepth. The syntax to be used is **CounterdrillDepth** =12mm.

Countersunk Hole

Definition:

A mechanical feature of Hole type you create when you click the  icon in the Part Design workbench. For more information, refer to the *Part Design User's Guide*.





Attributes:

A countersunk hole is defined by the following attributes:

- CountersinkAngle. The syntax to be used is **CountersinkAngle = 12deg**.
- CountersinkDepth. The syntax to be used is **CountersinkDepth** =15mm.
- CountersinkDiameter. The syntax to be used is **CountersinkDiameter** =15mm.

Cylinder

Definition:

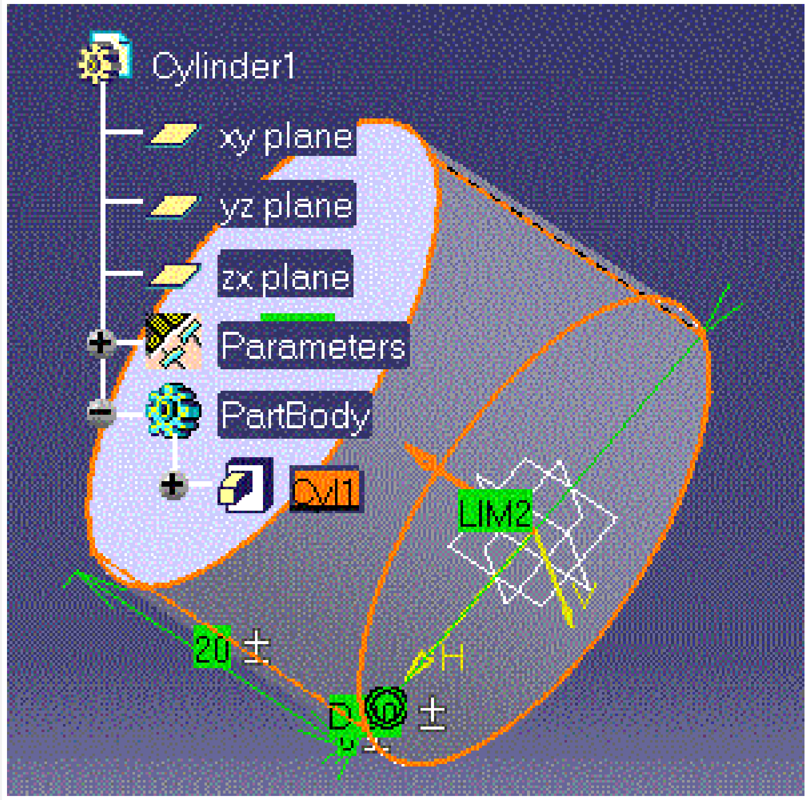
A cylinder is a pad created by extruding a circular sketch.

Attributes:

A cynlinder is defined by the following attributes:

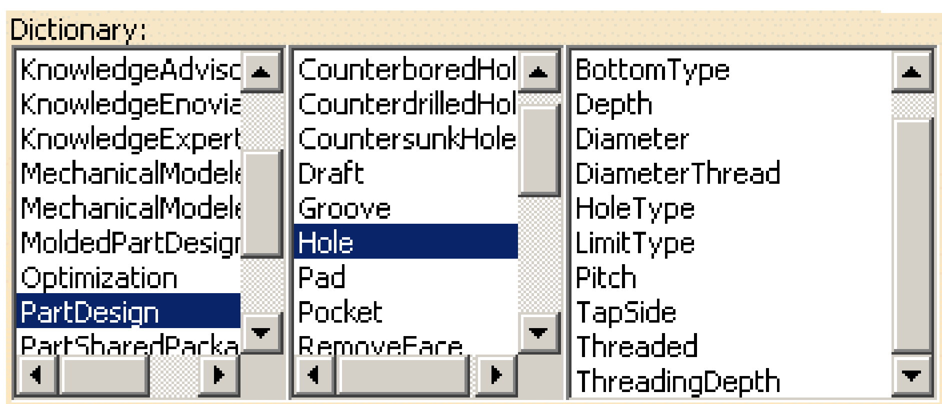
- EndLimit\Length. The syntax to be used is **Length =12mm**.
- Radius*:* The syntax to be used is **Radius =5mm**.

```
Cylinder1 isa CATPart
  {
    Part isa Part
     {
       PartBody isa BodyFeature
         {
           // Create a cylinder
           Cyl1 isa Cylinder
              {
                Radius=15.0 mm;
                EndLimit=20.0 mm;
              }
         }
     }
  }
```



Hole

Definition:



A is an opening through a feature.

Attributes:

A hole is defined by the following attributes:

- *BottomAngle*
- *BottomType*
- *Depth*
- *Diameter*
- *DiameterThread*
- *HoleType*
- *LimitType*
- *Pitch*
- *TapSide*
- *Threaded*
- *ThreadingDepth*
- Radius

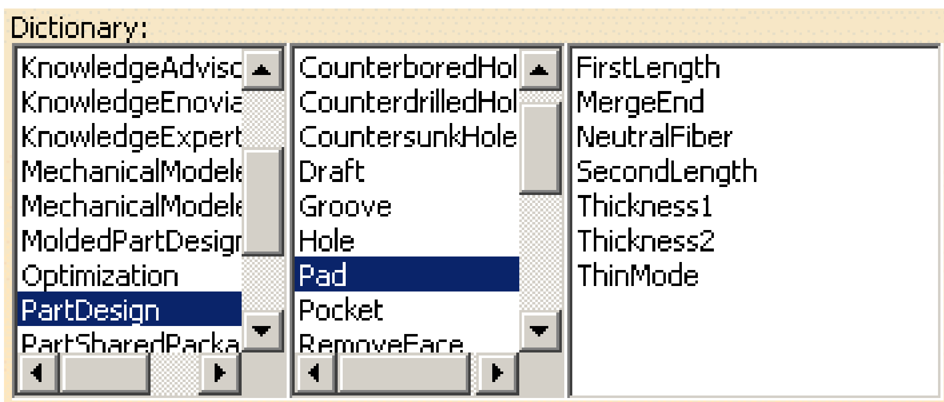
To specify a hole within your script, you have to use one of the holes listed below. Hole is the father type and cannot be used.

- Counterbored Hole
- Countersunk Hole
- Counterdrilled Hole
- Tapered Hole

Pad

Definition:

A pad is a feature created by extruding a sketch.



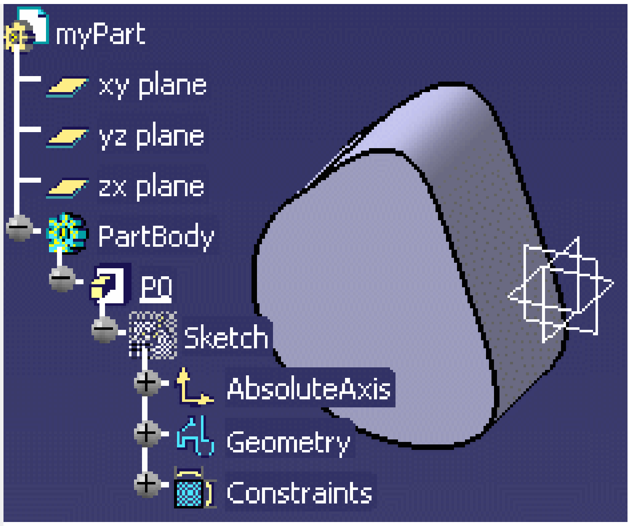
Attributes:

A pad is defined by the following attributes:

- *the sketch* the pad is extruded from.
- the *FirstLimit\Length* (or *StartLimit\Length*)
- the *SecondLimit\Length* (or *EndLimit\Length*).

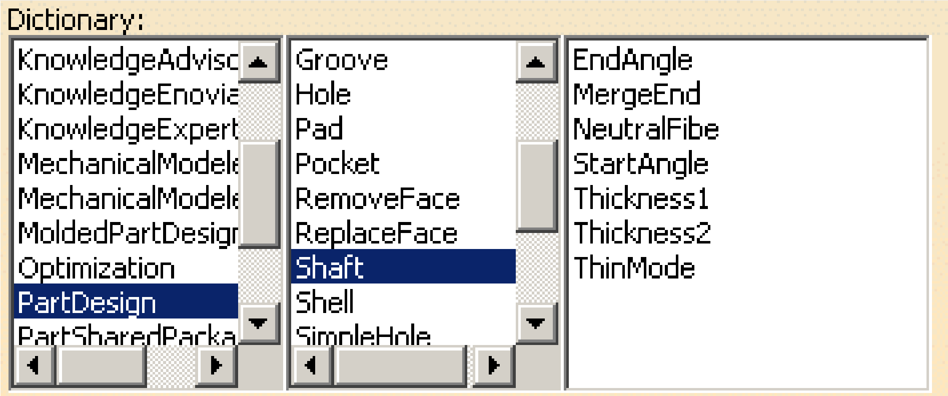
A limit which is not specified is set by default to zero.

```
// Use the Insert File Path command from the
// contextual menu to specify the path of the file
// to be imported

import  PktSketchToImport.CATPart;/*In the script
above, the P0 pad is created from the Sketch.1 sketch
which is imported from the  document.*/



myDocument isa CATPart
  {
    myPart isa Part
      {
        PartBody isa BodyFeature
          {
            Sketch isa Sketch.1
            {}
            P0 isa Pad("Sketch")
            {
             SecondLimit\Length=40.0mm;
            }
          }
      }
  }
```



Shaft

A shaft is a feature created by rotating a sketch around and axis.



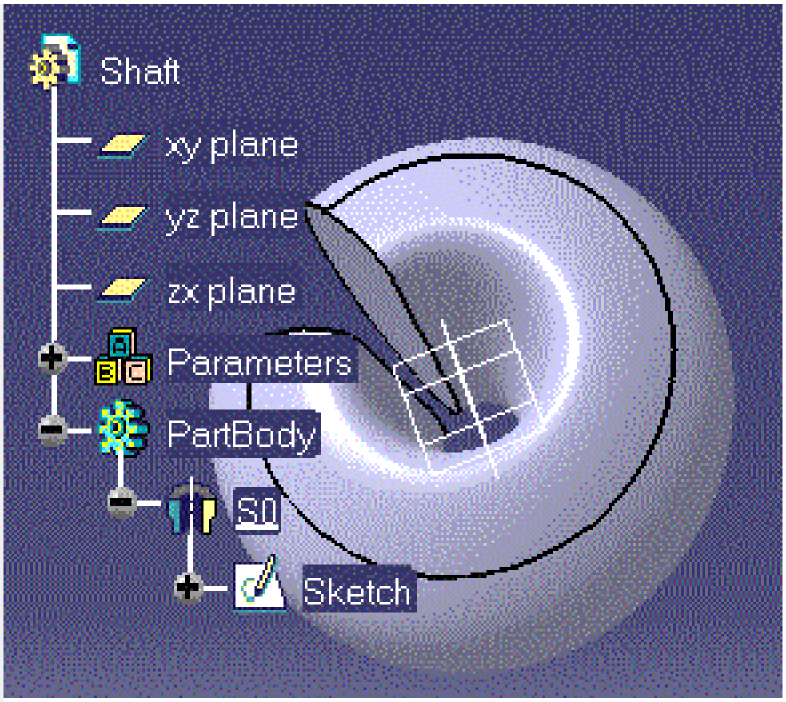
Attributes:

A shaft has two attributes:

- The *StartAngle*. The syntax to be used is **StartAngle=12deg**.
- The *EndAngle*. The syntax to be used is **EndAngle=23deg**.
- The MergeEnd
- The NeutralFiber
- The Thickness1
- The Thickness2
- The Thinmode

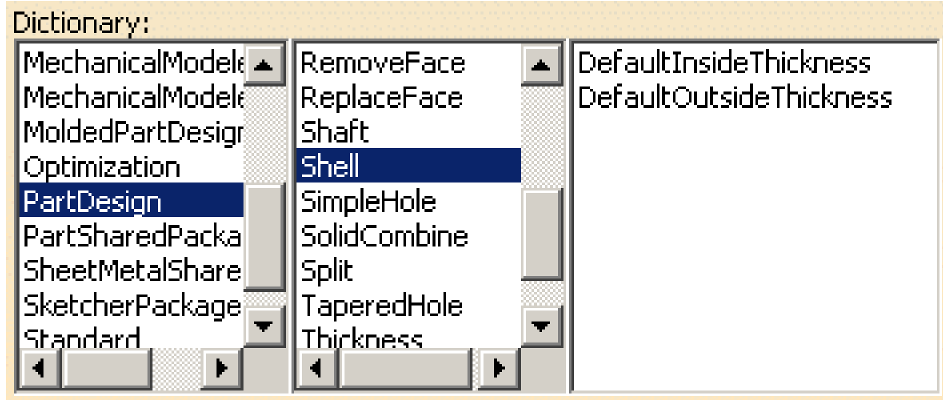
The sketch to be rotated must be imported from an external CATPart document. This external document must also include a rotation axis.

```
/* Use the Insert File Path command from the
contextual menu */
/* to specify the sketch to be imported */
import sketch to be imported. The file must contain
Sketch.1 */
import //Use the Insert File Path command to insert the
Pktsketch_shaft.CATPart file.
MyShaft isa CATPart
  {
    myPart isa Part
      {
        PartBody isa BodyFeature
          {
            Sketch isa Sketch.1 {}
            S0 isa Shaft("Sketch")
              {
                StartAngle = 20 deg ;
                EndAngle = 300 deg ;
              }
          }
      }
  }
```



Shell

Definition:



A shell is a hollowed out feature.

Attributes:

A shell is defined by the following attributes:

- DefaultInsideThickness. The syntax to be used is **DefaultInsideThickness = 2mm**.
- DefaultOutsideThickness*:* The syntax to be used is **DefaultOutsideThickness = 1mm**.

To specify a shell within your script, you must have a part open, then:

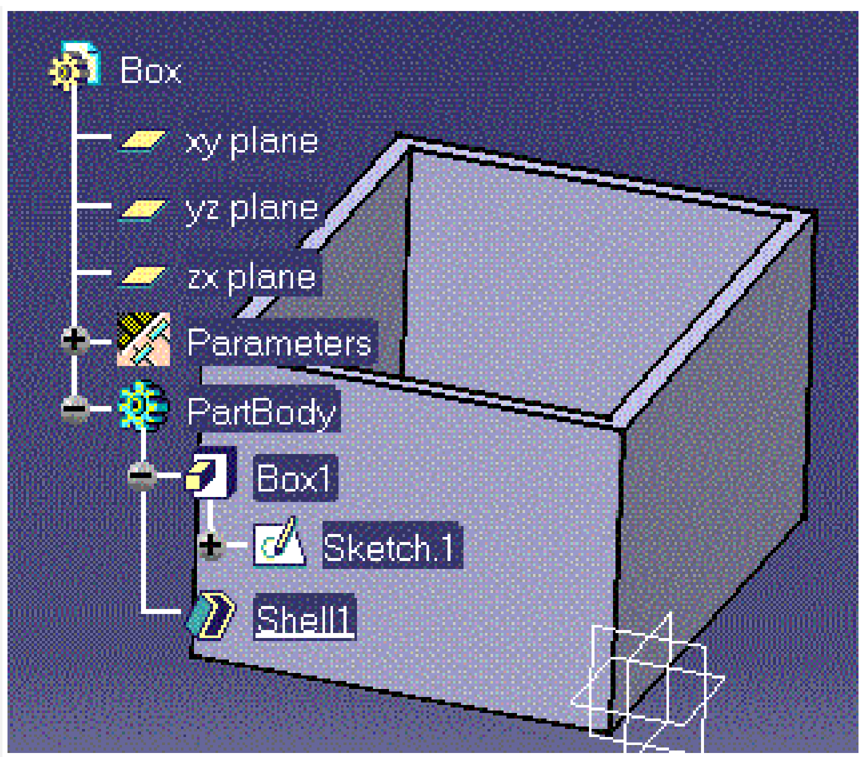
1. create a Shell by using the isa function

   Shell1 isa Shell ( ) { }

2. right-click anywhere inside the parentheses and select the 'Get Surface' function from the contextual menu. Then, in the geometry area, select the face to be hollowed out.

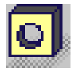
A 1mm thick shell is created by default.

```
MyBox isa CATPart
 {
   BoxPart isa Part
    {
      PartBody isa BodyFeature
       {
        Box1 isa Box
         {
          Width = 20.0 mm;
          Height = 25.0 mm;
          Length = 15.0 mm;
         }
      Shell1 isa Shell (face definition)
         {
          DefaultInsideThickness = 2mm;
          DefaultOutsideThickness = 1mm;
         }
       }
    }
}
```



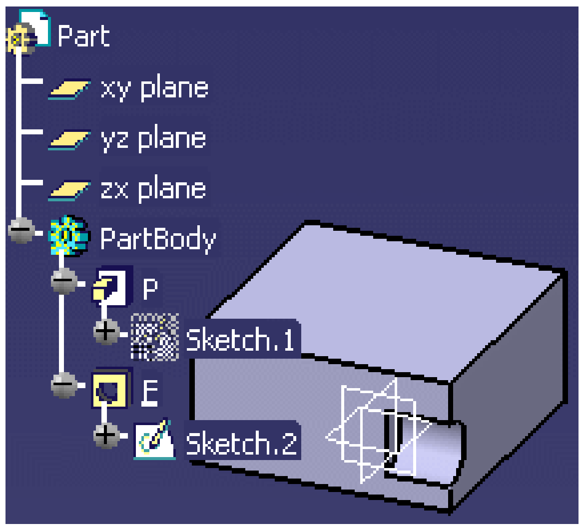
SimpleHole

Definition:

A mechanical feature of Hole type you create when you click the  icon in the Part Design workbench. For more information, refer to the *Part Design User's Guide*..

Attributes:

```
Hole1 isa CATPart
{
   Part isa Part
     {
        PartBody isa BodyFeature
        {
            P isa Pad
              {
              }
F isa SimpleHole("Use the Get Edge command to
select the edge")
             {
             }
        }
     }
}
```



Sphere

Definition:

A sphere is a shaft created by rotating half a circle around an axis passing through the arc extremities. The only property is the Radius.

Attributes:

A sphere is defined by the following attribute:

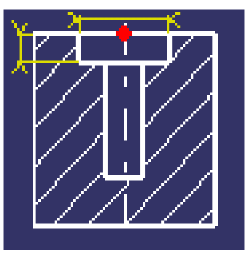
- Radius. The syntax to be used is: **Radius = 20.0 mm**.

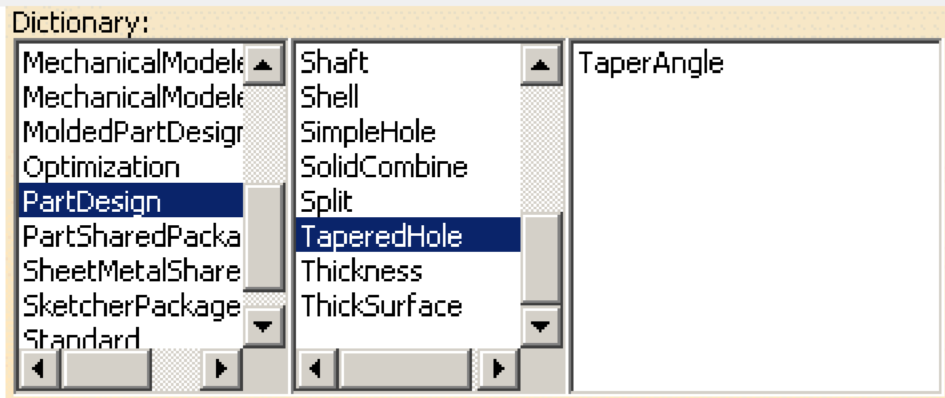
```
MySphere  isa CATPart
 {
   SpherePart isa Part
    {
      PartBody isa BodyFeature
       {
         Sphere1 isa Sphere
          {
           Radius = 20.0 mm ;
          }
       }
    }
}
```

Tapered Hole

Definition:

A mechanical feature of Hole type you create when you click the  icon in the Part Design workbench. For more information, refer to the *Part Design User's Guide*.





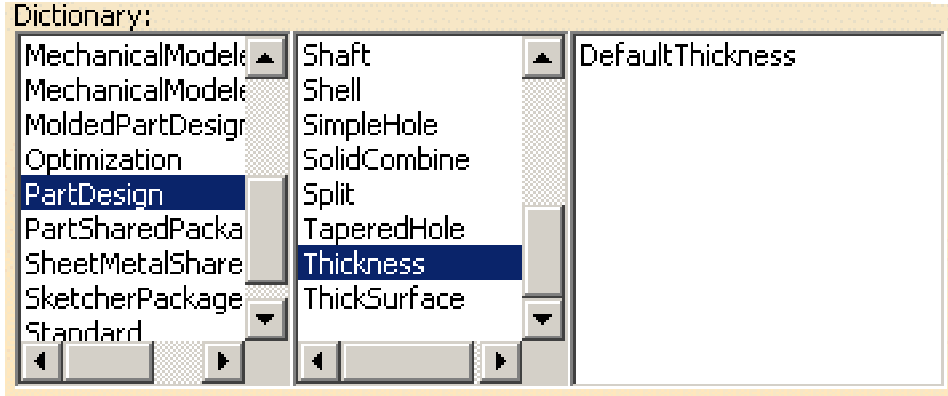
Attributes

A tapered hole is defined by the following attribute:

- TaperAngle:

Thickness

Definition:



A thick

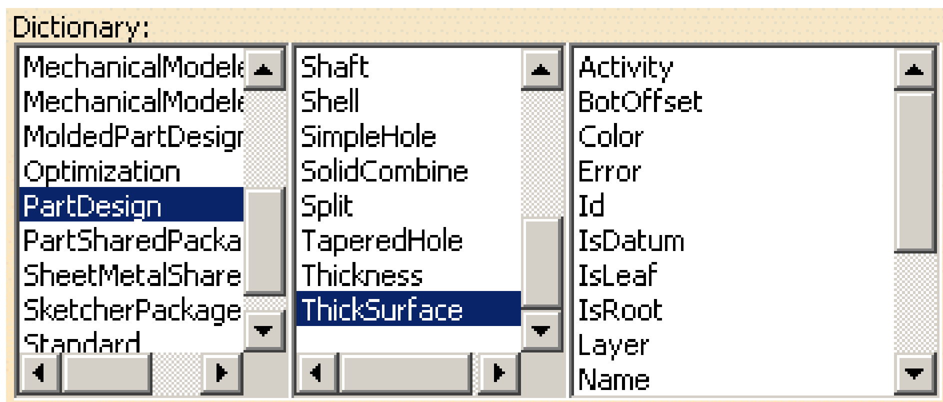
Attributes

A thickness is defined by the following attribute:

- DefaultThickness:

ThickSurface

Definition:



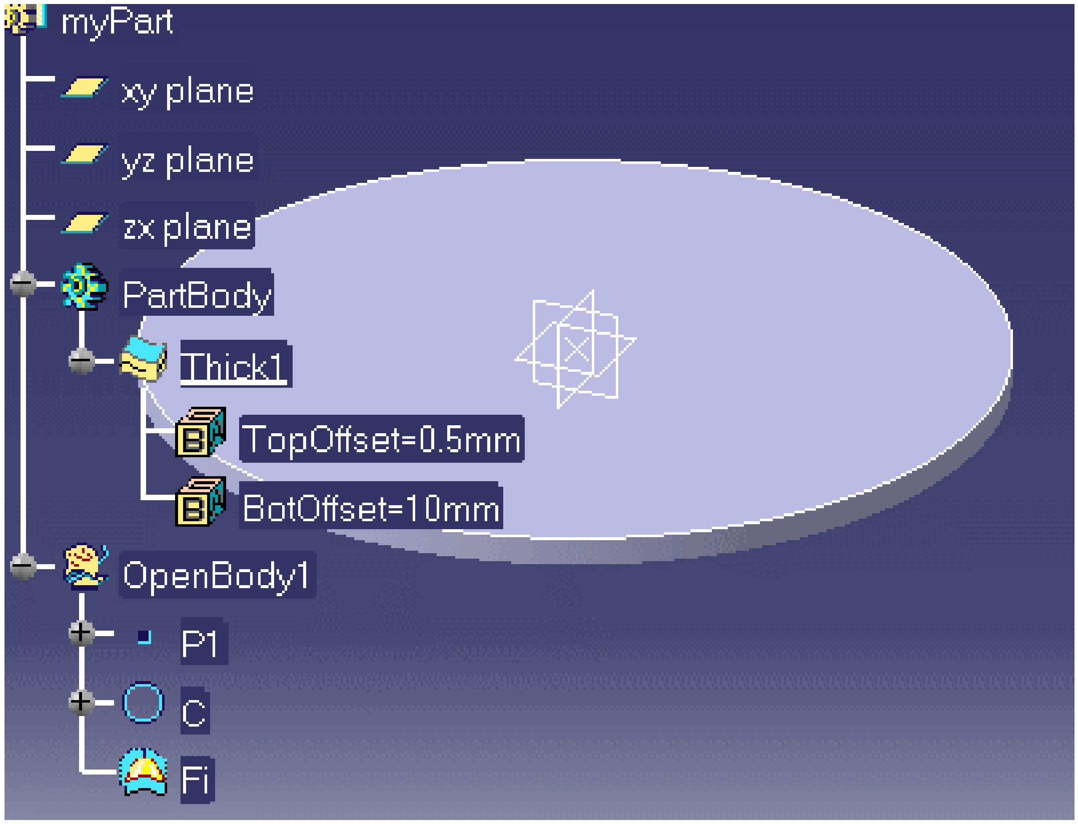
A thicksurface is a surface to which material was added in two opposite directions.

Attributes:

A thicksurface is defined by the following attributes:

- *TopOffset*, the thickness in one direction. The syntax to be used is **TopOffset = 0.5mm**.
- the *BotOffset*, the thickness in the one direction. The syntax to be used is **BotOffset = 10 mm**.

```
myThickSurface isa CATPart
  {
    myPart isa Part
       {
 OpenBody1 isa OpenBodyFeature
   {
            P1 isa GSMPoint
              {
                PointType = 0;
                TypeObject isa
GSMPointCoord
                    {
                      X = 0mm;
                      Y = 0mm;
                      Z = 0mm;
                    }
                }
            C isa GSMCircle
               {
                 CircleType = 0;
                 TypeObject isa
GSMCircleCtrRad
                    {
                       Center = object :
..\..\P1;
                       Support = object :
..\..\..\`xy-plane`;
                       Radius = 150mm;
                    }
                  StartAngle = 0deg;
                  EndAngle = 360deg;
                }
            Fi isa GSMFill
                {
                  Boundary = object :
..\C;
                }
     }

          PartBody isa BodyFeature
       {
             Thick1 isa ThickSurface
    {
    TopOffset = 0.5mm;
    BotOffset = 10 mm;
    Surface = object :
..\..\OpenBody1\Fi;
          }
}
    }
}
```



Torus

Definition:

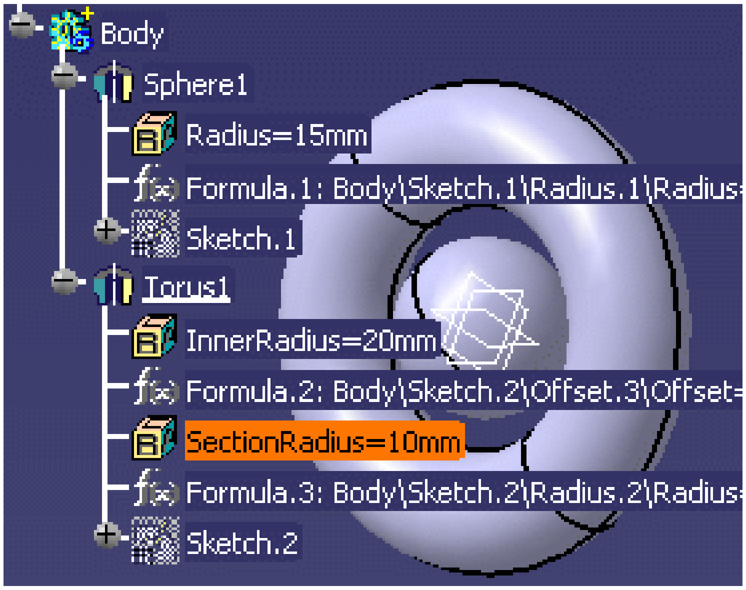
A torus is a shaft created by rotating a circular sketch around an axis.

Attributes:

A torus is defined by the following attributes:

- *InnerRadius*
- *SectionRadius*

```
BodyDoc isa CATPart
  {
  BodyPart isa Part
    {
     Body isa BodyFeature
       {
       // Create a sphere
       Sphere1 isa Sphere
           {
           Radius = 15.0 mm;
           }
        // Create a torus
       Torus1 isa Torus
        {
        InnerRadius = 20.0 mm ;
        SectionRadius = 10.0 mm ;
        }
      }
    }
}
```



Part Shared Package

ConstantEdgeFillet

Definition

A fillet is a curved surface of a constant or variable radius that is tangent to, and that joins two surfaces. Together, these three surfaces form either an inside corner or an outside corner.

Important Note:

To specify a fillet within your script, you must have a part open, then:

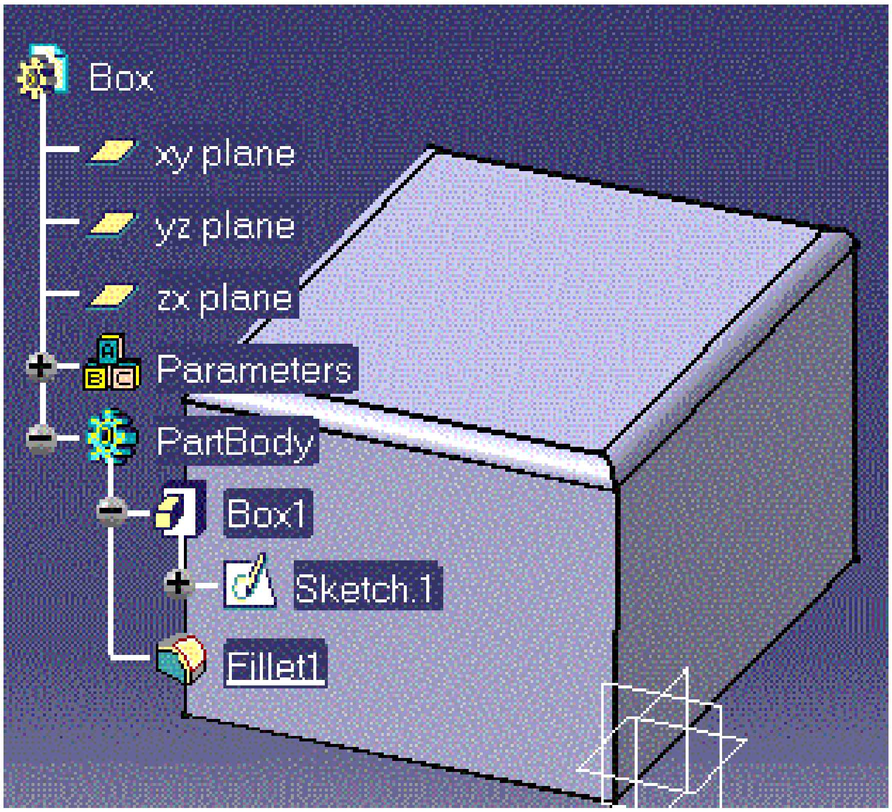
1. Create a Fillet by using the isa keyword.

   Fillet1 isa ConstantEdgeFillet ( ) { }

2. Right-click anywhere inside the parentheses and select the 'Get Edge' or the 'Get Surface' function from the contextual menu. Then, in the geometry area, select the edge or the face to be filleted.

Example

```
Box1 isa CATPart
  {
    BoxPart isa Part
       {
        PartBody isa BodyFeature
          {
            Box1 isa Box
              {
                Width = 20.0 mm ;
                Height = 25.0 mm ;
                Length = 15.0 mm ;
              }
       // Use the Get Edge or Get Surface command
       // from the contextual menu to retrieve
       // the edge or face to be filleted
             Fillet1 isa ConstantEdgeFillet ( face to be filleted)
              {
                Radius =1.0 mm;
              }
           }
       }
  }
```



Fillet

Definition

Describes the feature you create when you click the  icon in the Part Design workbench. For more information, please refer to the *Part Design User's Guide*. It is defined by one property:

- *Radius*

There are 3 different types of fillets:

- ConstantEdgeFillet
- FaceFillet
- TriTangentFillet

Pattern

Definition

A pattern is a set of similar features repeated in the same part. Two types of patterns can be created with *CATIA*: the rectangular patterns and the circular patterns. At present, only rectangular patterns can be generated from a script. A rectangular pattern is defined by the following properties:

- Nb1, the number of elements to be replicated along the first direction
- Nb2, the number of elements to be replicated along the second direction
- Step1, the element spacing along the first direction
- Step2, the element spacing along the second direction
- Activity.

Syntax

pattern1 isa pattern [*Nb1*,*Nb2*] of *feature_to_be_repeated*

Example

```
MyBox isa CATPart
{
BoxPart isa Part
{
PartBody isa BodyFeature
{
Box1 isa Box
{
Width = 20 mm ;
Height = 20 mm ;
Length = 20 mm ;
}

// Use the Get Surface command from the
// contextual menu to specify the hole
// anchor
Hole1 isa SimpleHole ("Face:(Brp:(Pad.1;2);None:();Cf9:())")
{
Diameter = 15 mm;
}
Pattern1 isa Pattern[3,4] of Hole1
{
Step1 = 50 mm;
Step2 = 50 mm;
}
}
}
```

Standard Package

Old Types Names	Old Attributes	New Types Names	New Attributes
-	-	Feature	Id Name Owner
-	-	List	-
-	-	Visualizable	Color Layer Pick Show

GSD Shared Package

Types Names	Attributes
GSMAffinity	AxisFirstDirection AxisOrigin AxisPlane Ratio
GSMAxisToAxis	
GSMRotate	Angle Axis
GSMScaling	Ratio Reference
GSMSymetry	Reference
GSMTransformation	Activity ToTransfor
GSMTranslate	Direction Distance

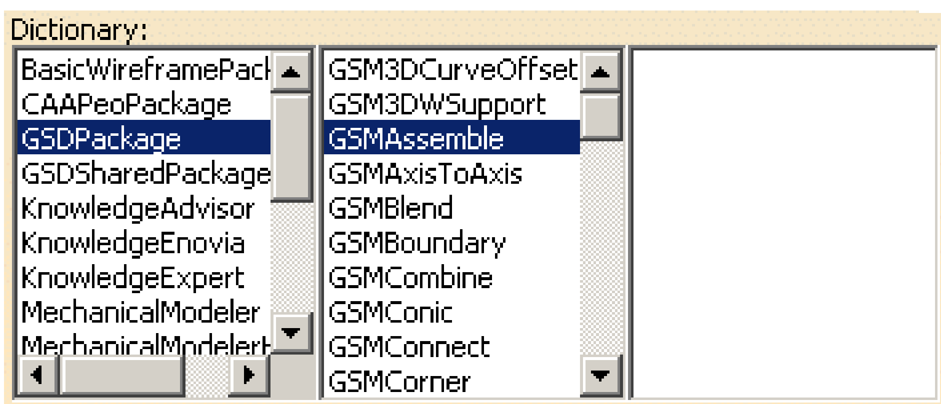
GSD Package

Please find below a table listing the types contained in the GSD package:

GSM3DCurveOffset	GSMAssemble	*GSMAxisToAxis*
GSMBlend	GSMBoundary	GSMCombine
GSMConic	GSMConnect	GSMCorner
GSMCurve	GSMCurvePar	GSMCurveSmooth
GSMCylinder	GSMDirection	GSMExtract
GSMExtractContour	GSMExtrapol	GSMExtremum
GSMExtremumPolar	GSMExtrude	GSMFill
GSMFillet	*GSMFilletBiTangent*	GSMHealing
GSMHelix	GSMIntersect	GSMInverse
GSMLawDistProj	*GSMLineCorner*	GSMLoft
GSMNear	GSMOffset	GSMProject
GSMReflectLine	GSMRevol	GSMSphere
GSMSpine	GSMSpiral	GSMSplit
GSMSweep	GSMSweepCircle	GSMSweepConic
GSMSweepSegment	*GSMSweepSketch*	GSMTrim
GSMUnfold	*GSMWSupport*	GSOBump
GSOJunction	GSOSeatDiabolo	GSPShapeMorphing
GSOVariableOffset	GSOWrapCurve	GSOWrapSurface

GSMAssemble

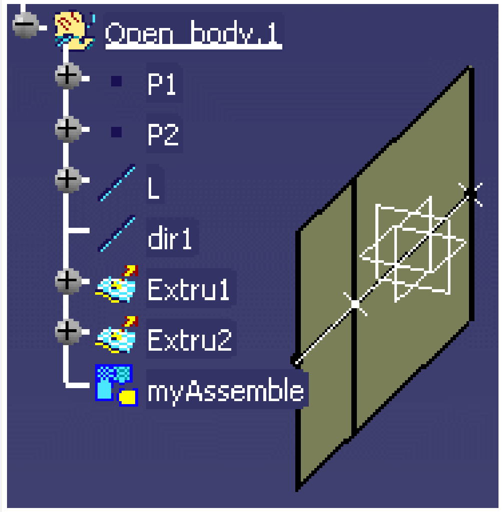
Definition:



A GSMAssemble is an object which joins at least two surfaces or two curves. The surfaces or curves to be joined must be adjacent. See the *Generative Shape Design User's Guide* for more information.

Attributes:

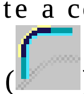
Click here to open the GSMAssembleScript script file.

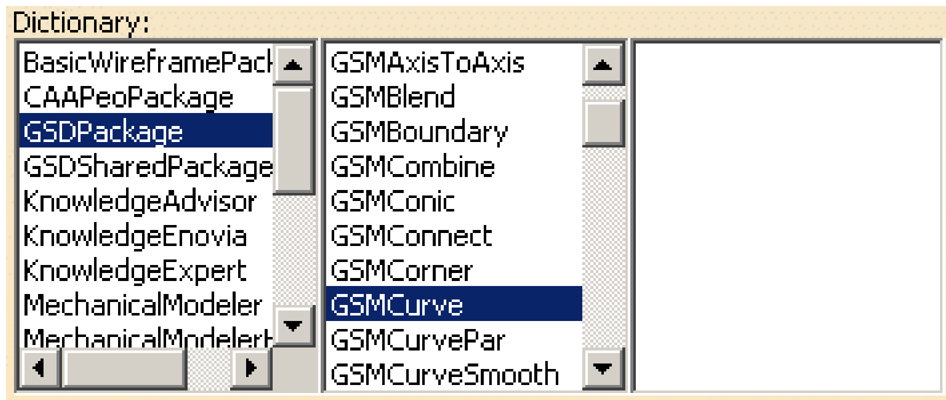


GSMCurve

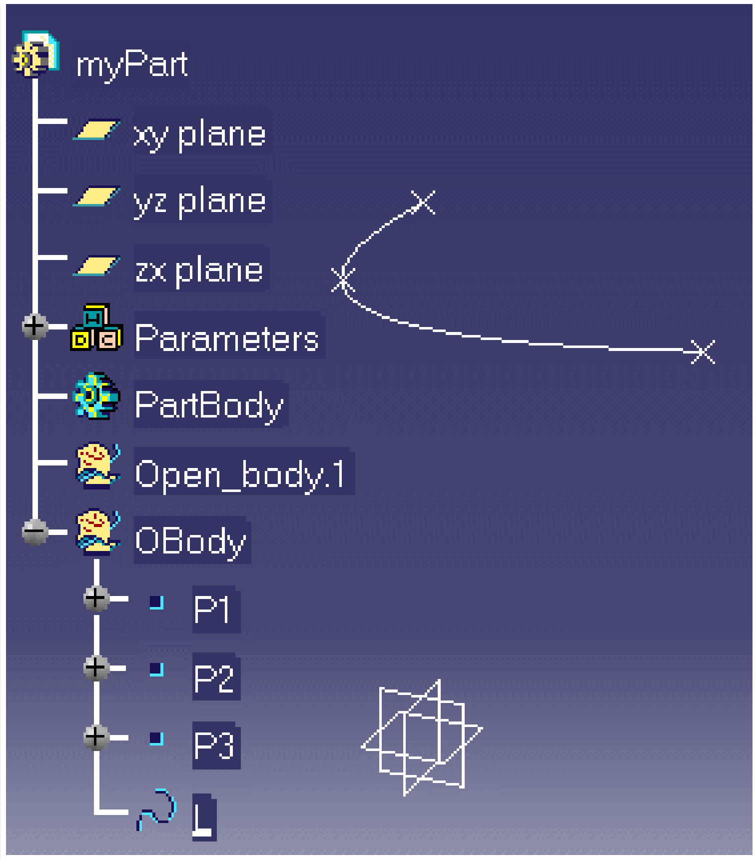
Definition:

A GSMCurve is an object generated by the Generative Shape Design product.

You can create a corner by clicking the **Corner** icon () in the Generative Shape Design workbench.



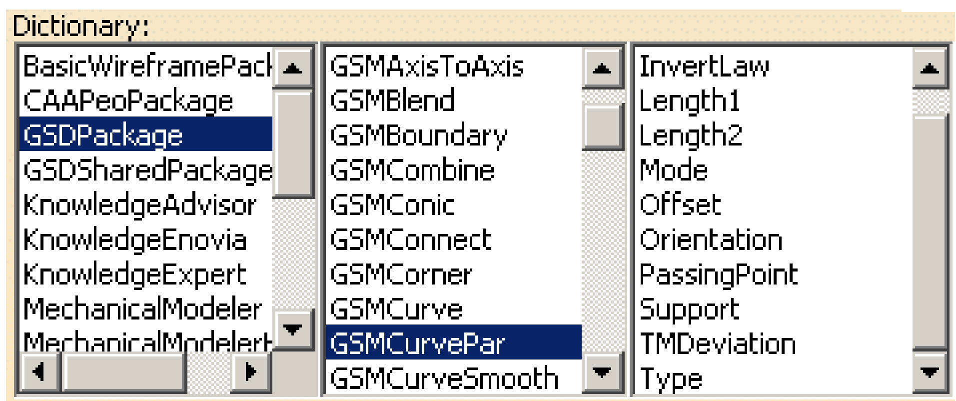
Click here to open the GSMCurveScript script file.



GSMCurvePar

Definition:

An GSMCurvePar object is a Generative Shape Design parallel curve.

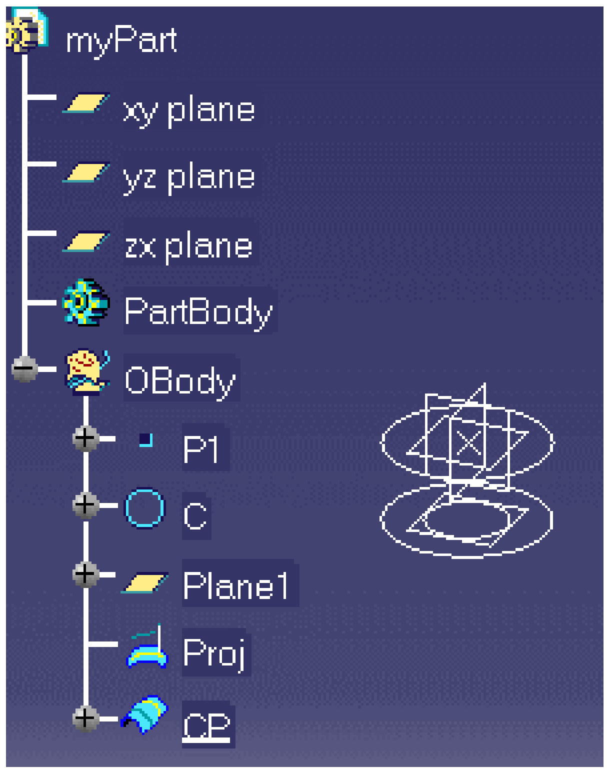


Attributes:

A GSMCurvePar is defined by the following attributes:

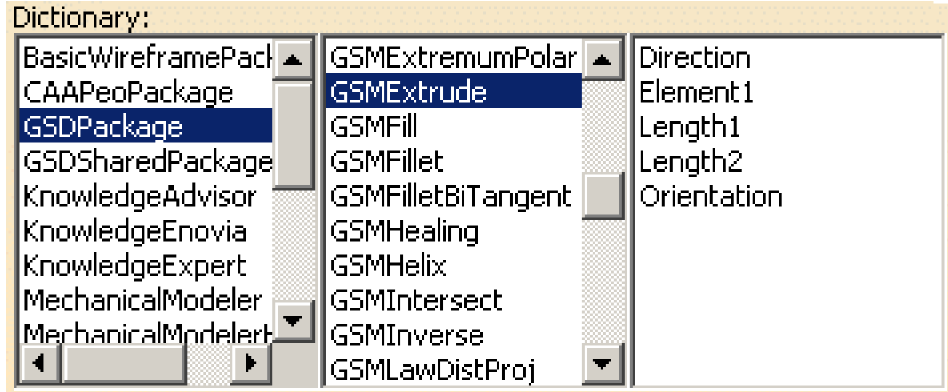
- *InvertLaw*.
- *Length1*.
- *Length2*.
- *Mode*.
- *Offset*.
- *Orientation*.
- *PassingPoint*.
- *Support*.
- *TMDeviation*.
- *Type*.

Click here to open the GSMCurveParScript script file.



GSMExtrude

Definition:



A surface created by extruding a profile along a given direction.

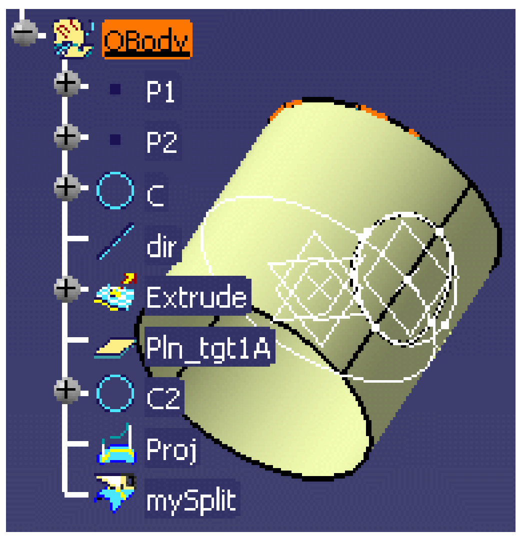
You can create an extruded surface by clicking the **Extrude** icon () in the Generative Shape Design workbench.

Attributes:

A GSMExtrude is defined by the following attributes:

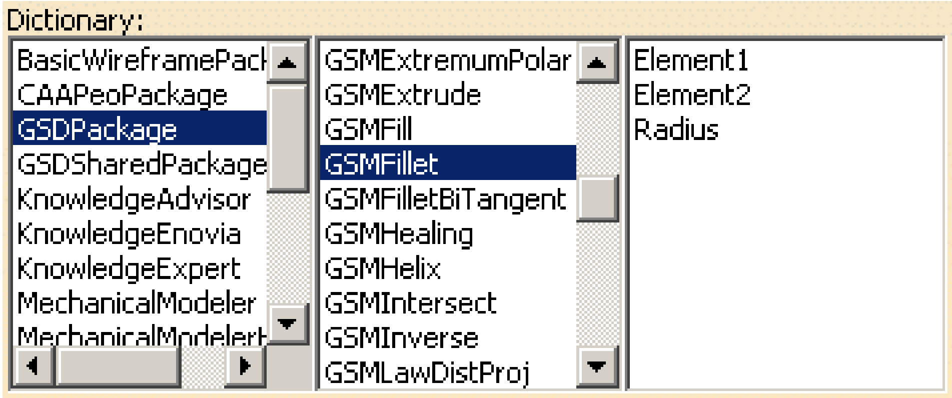
- *Direction*.
- *Element1*.
- *Length1*.
- *Length2*.
- *Orientation*.

Click here to open the GSMSplit script file.



GSMFillet

Definition:



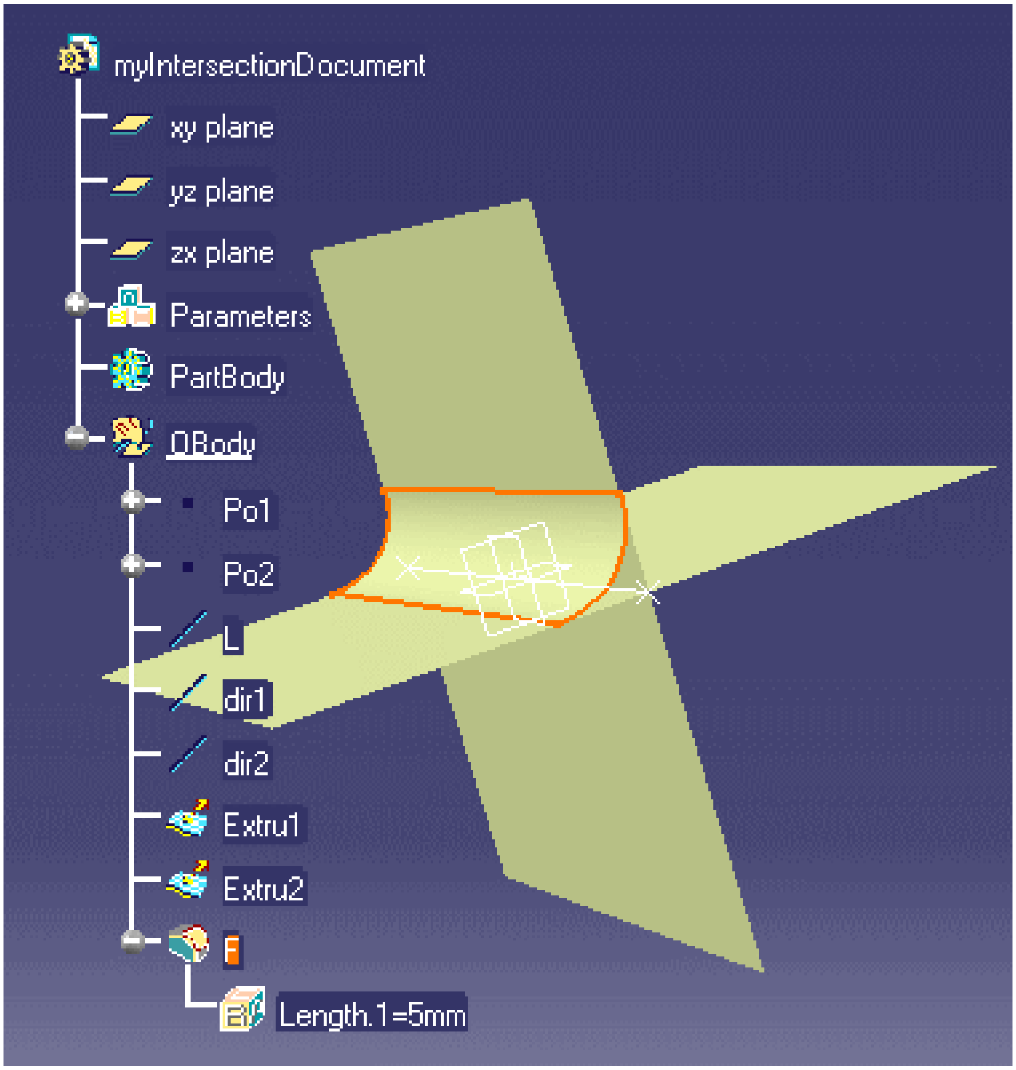
An GSMFillet object is curved surface of a constant or variable radius that is tangent to and joins two surfaces. Together these three surfaces form either an inner or outer corner.

Attributes:

A GSMFillet is defined by the following attributes:

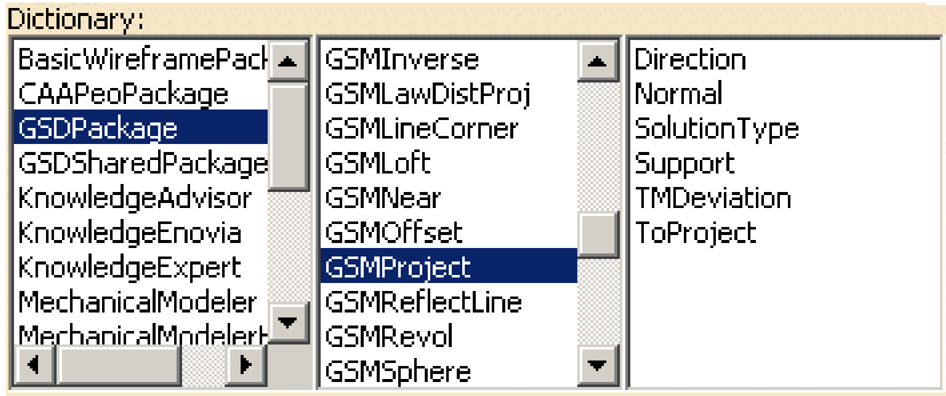
- *Element1*.
- *Element2*.
- *Radius*.

Click here to open the GSMFilletScript.CATGScript file.



GSMProject

Definition:



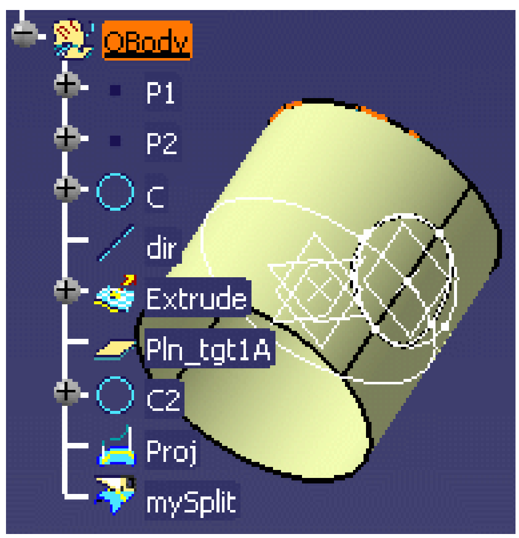
A Generative Shape Design projection. See the *Generative Shape Design User's Guide* for more information.

Attributes:

A GSMProject is defined by the following attributes:

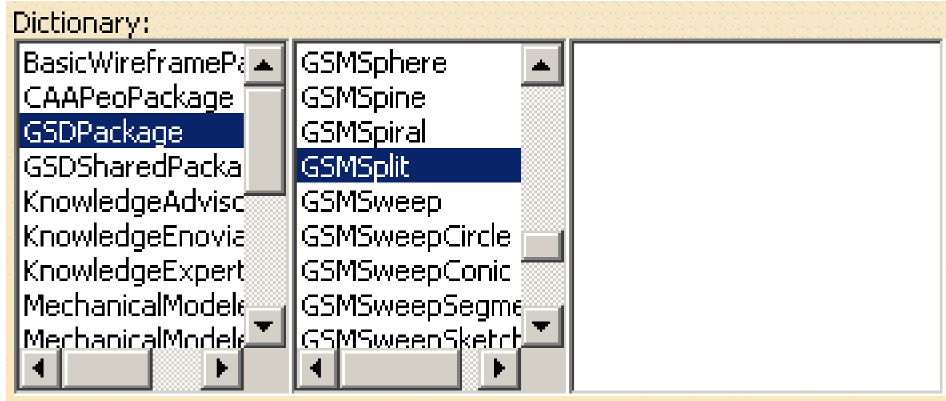
- *Direction*.
- *Normal* which corresponds to the Projection type field in the Projection Definition dialog box (Normal = 1 for an orthogonal projection - otherwise specify a direction, a GSMLine for example).
- *SolutionType*.
- *TMDeviation*.
- *ToProject* which corresponds to the Projected field in the Projection Definition dialog box.
- *Support* which corresponds to the Support field in the Projection Definition dialog box.

Click here to open the GSMSplit script file.



GSMSplit

Definition:

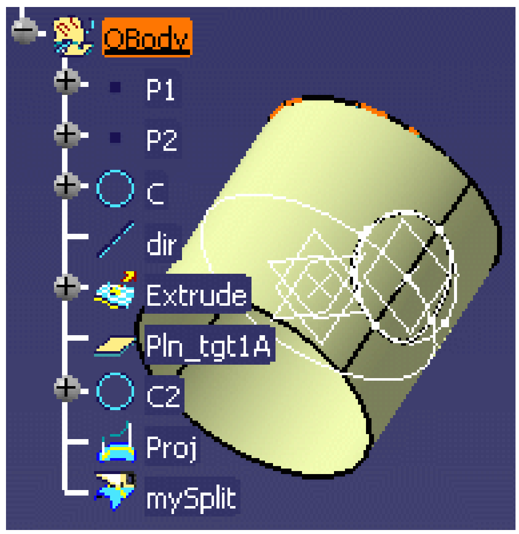


A surface or wireframe element that was split by means of a cutting element. You can split:

- a wireframe element by a point, another wireframe element or a surface
- a surface by a wireframe element or another surface.

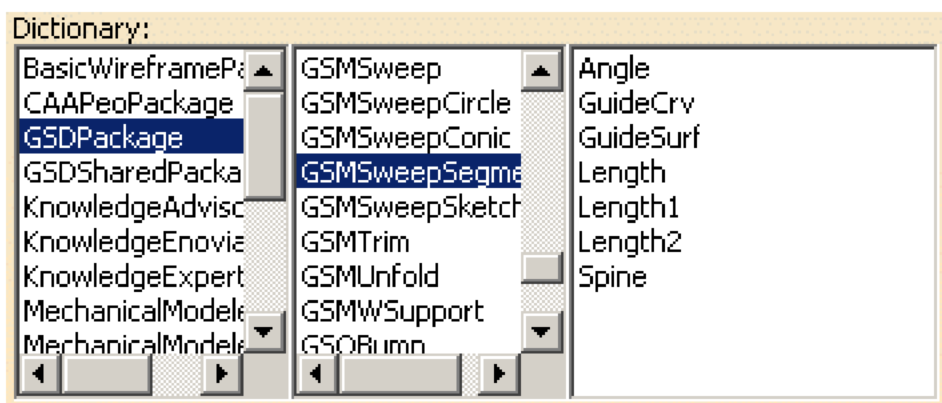
You can split geometry by clicking the **Split** icon () in the Generative Shape Design workbench.

Click here to open the GSMSplit script file.



GSMSweepSegment

Definition:



Attributes:

A GSMSweepSegment is defined by the following attributes:

- *Angle*.
- *GuideCrv*.
- *GuideSurf*.
- *Length*.
- *Length1*.
- *Length2*.
- *Spine*.

Click here to open the GMSweepSegmentScript file.

Knowledge Expert

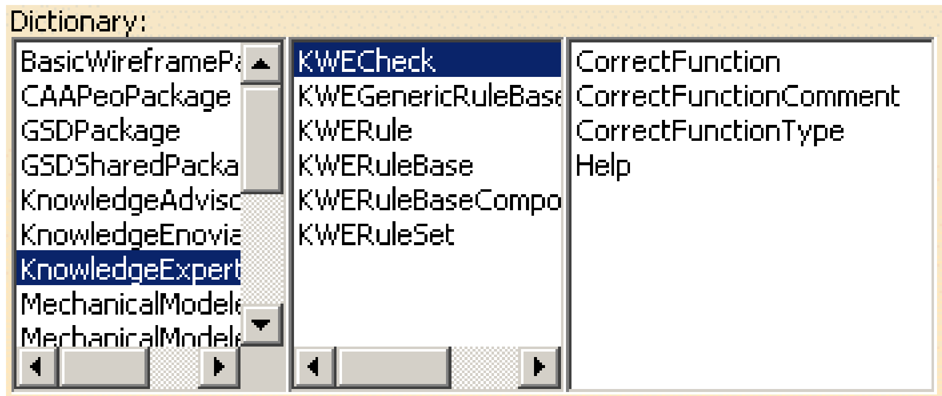
Please find below a table listing the types contained in the Knowledge Expert package:

KWECheck	*KWEGenericRuleBaseComponent*	KWERule
KWERuleBase	KWERuleSet	*KWERuleBaseComponent*

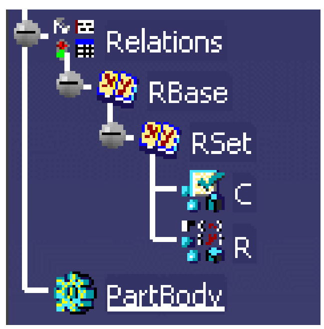
KWECheck

Definition:

Expert Checks are features generated by the Knowledge Expert product. Checks are regrouped into rule sets. Rule sets belong to a rule base. When writing a script with checks you must comply with the Rule Base/Rule Set hierarchy. Refer to the *Knowledge Expert User's Guide* for more information on the concepts behind the expert rules and checks.



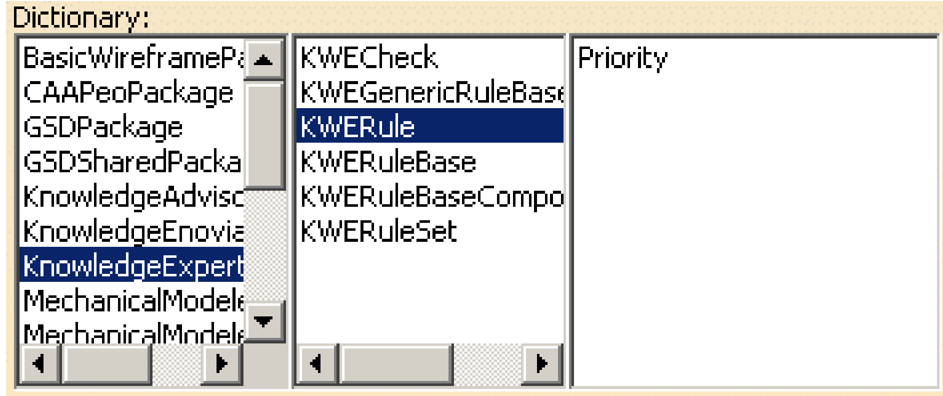
Click here to open the KnowledgeExpertScript file.



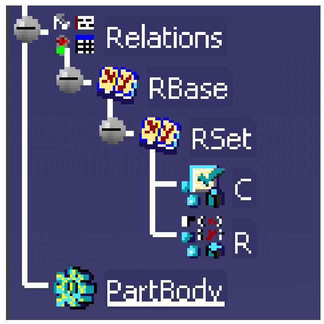
KWERule

Definition:

Expert Rules are features generated by the Knowledge Expert product. Rules are regrouped into rule sets. Rule sets belong to a rule base. When writing a script with rules you must comply with the Rule Base/Rule Set hierarchy. Refer to the *Knowledge Expert User's Guide* for more information on the concepts behind the expert rules and checks.



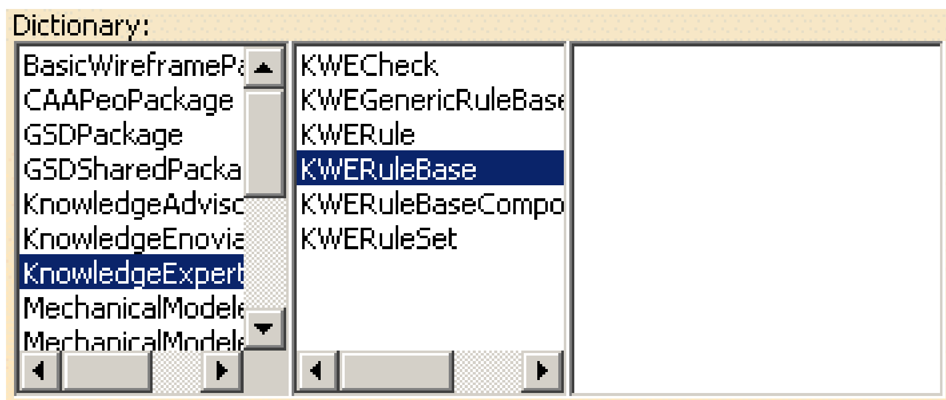
Click here to open the KnowledgeExpertScript file.



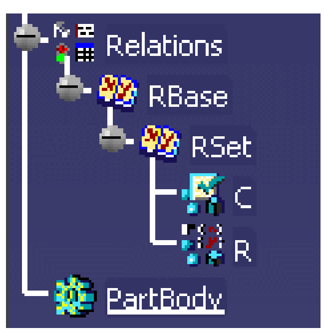
KWERuleBase

Definition:

Rule bases are features generated by the Knowledge Expert product. Refer to the *Knowledge Expert User's Guide* for more information on the concepts behind this type of feature.

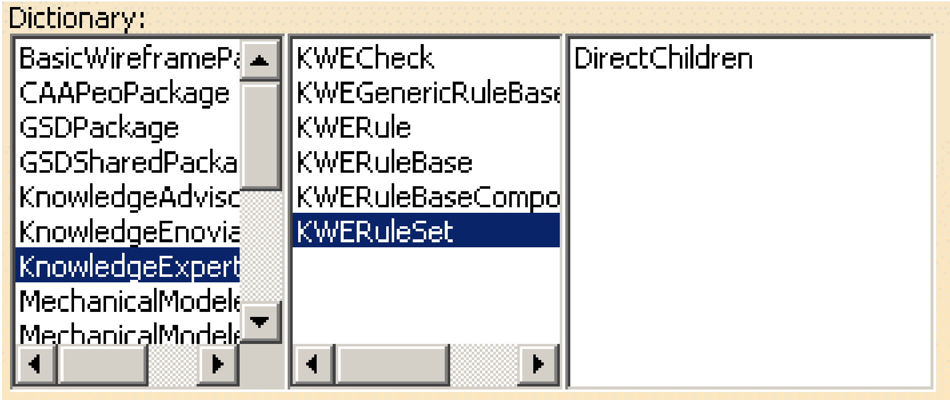


Click here to open the KnowledgeExpertScript file.



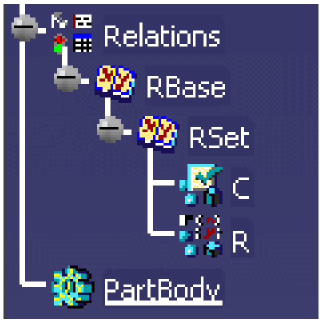
KWERuleSet

Definition:



Rule sets are features generated by the Knowledge Expert product. Refer to the *Knowledge Expert User's Guide* for more information on the concepts behind this type of feature.

Click here to open the KnowledgeExpertScript file.



Mechanical Modeler

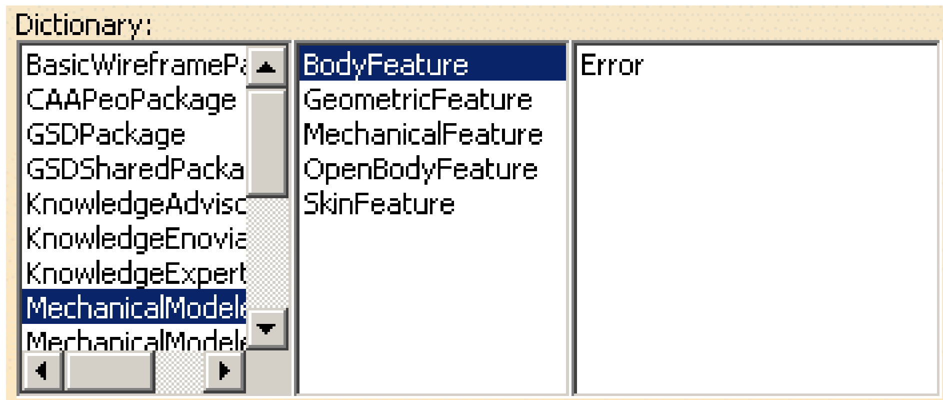
Some types and attributes were changed. Please find below a conversion table listing the old types, their attributes, their new names (if any) as well as their attributes:

BodyFeature	*GeometryFeature*
MechanicalFeature	*OpenBodyFeature*
OpenBodyFeature	

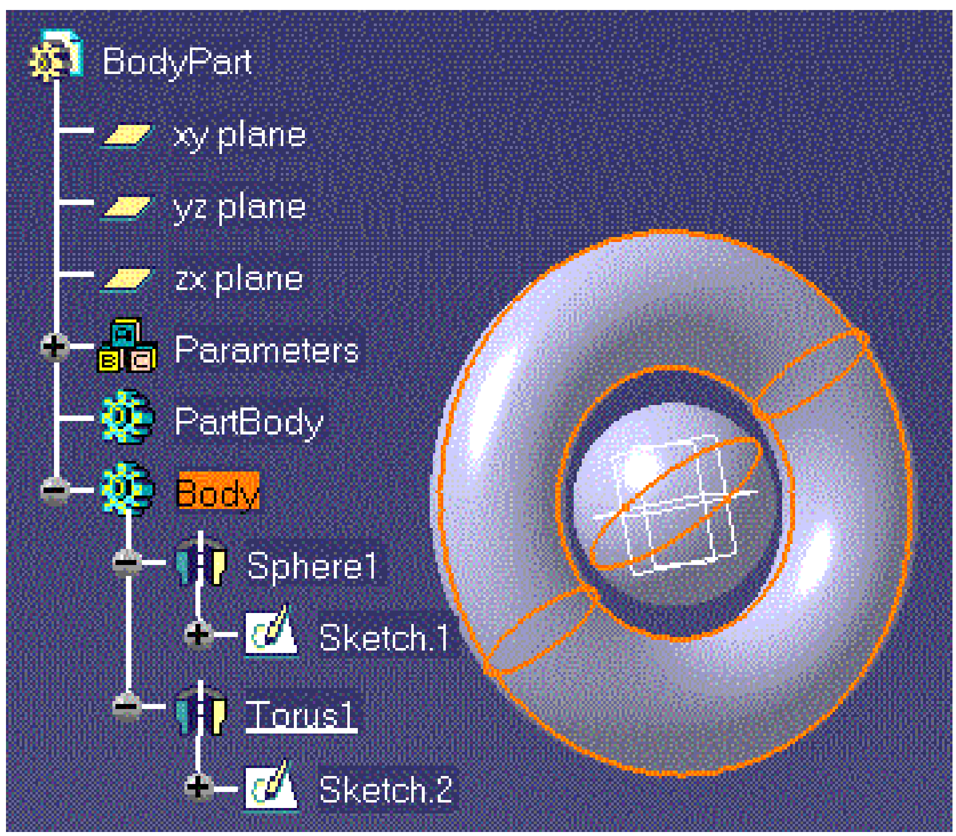
BodyFeature

Definition:

A body is the combination of several features within a part. For more information, see the *Part Design User's Guide*.



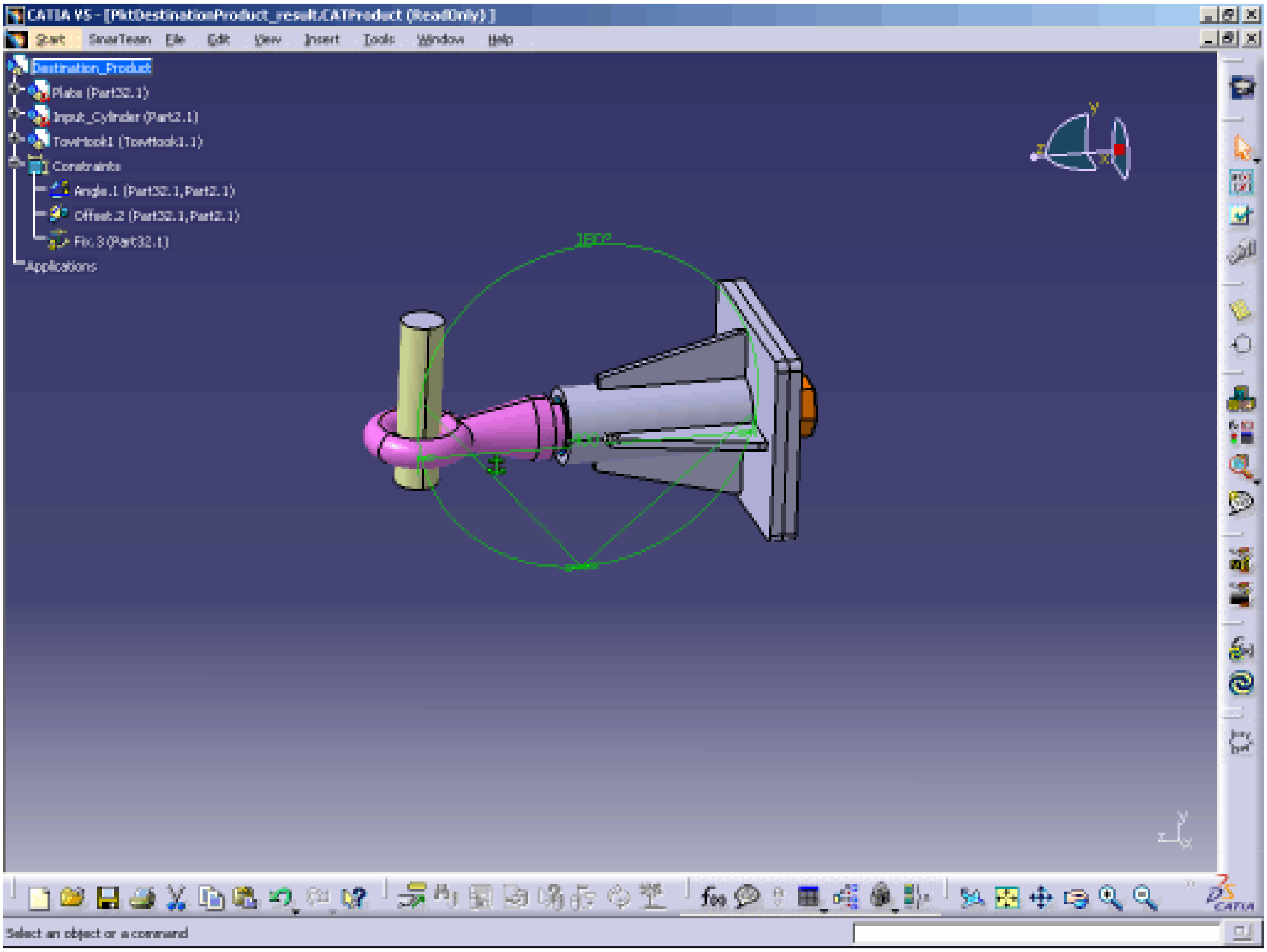

```
BodyDoc isa CATPart
  {
   BodyPart isa Part
     {
       Body isa BodyFeature
         {
           // Create a sphere
           Sphere1 isa Sphere
             {
               Radius = 15.0 mm;
             }
           // Create a torus
           Torus1 isa Torus
             {
               InnerRadius = 20.0 mm ;
               SectionRadius = 10.0 mm ;
             }
         }
     }
  }
```



Workbench Description

This section contains the description of the icons and menus specific to the Knowledge Advisor workbench.

The Knowledge Advisor workbench is shown below. Click the sensitive areas (toolbars) to access the related documentation.



Knowledge Advisor Menu Bar

The various menus and menu commands that are specific to Knowledge Advisor are described below.

Start	File	Edit	View	Insert	Tools	Window	Help

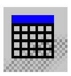
Tools

For	See
Formula...	Creating a Formula

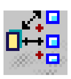
Knowledge Toolbar

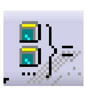
The Knowledge toolbar contains the following tools

 See Creating a Parameter and Creating a Formula.

 See Creating a Design Table from Current Values and Creating a Design Table from a Pre-Existing File.

 See Creating a Law.

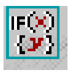
 See Using the Knowledge Inspector.

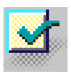
 See Using the Equivalent Dimensions Feature.

 See Locking and Unlocking a Parameter.

Reactive Features Toolbar

The Reactive Features Toolbar contains the following tools:

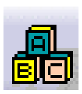
 See Creating a Rule.

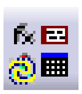
 See Creating a Check.

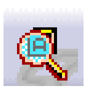
 See Creating a Reaction: DragAndDrop Event.

Organize Knowledge Toolbar

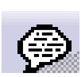
The Organize Knowledge Toolbar contains the following tools:

 See Creating Sets of Parameters.

 See Creating Sets of Relations.

 See Adding a Parameter to a Feature.

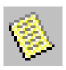
 See Adding a Parameter to an Edge.

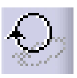
 See Associating URLs and Comments with Parameters or Relations.

Control Features Toolbar

The Control Features Toolbar contains the following tools:

Here is a brief description of each icon.

 See Creating a List.

 See Creating a Loop.

Actions Toolbar

The Control Features Toolbar contains the following tools:

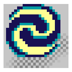
 See Launching a VB macro with Argument.

 See Using the Knowledge Advisor Action Feature.

Tools Toolbar

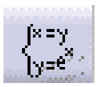
The Tools Toolbar contains the following tools:

 See Updating Relations Using Measures.

 Select the Update icon to update a document without exiting the Knowledge Advisor workbench.

Set of Equations Toolbar

The Tools Toolbar contains the following tools:

 See Solving a Set of Equations.

Customizing

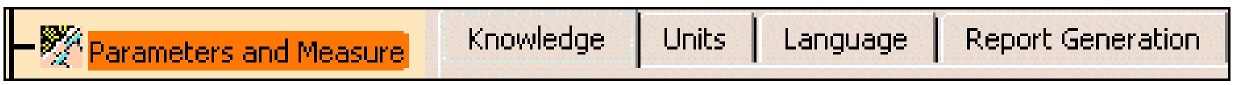
Before you start your first working session, you can customize the way you work to suit your habits.
This type of customization is stored in permanent setting files: these settings will not be lost if you end your session

1. Select the **Tools -> Options** command. The **Options** dialog box displays.

   The Options dialog box displays.

2. Choose the **General** category in the left-hand box.

3. Click the **Parameters and measure** tab. The following tabs display.



This tab lets you define:

- the Knowledge settings
- the libraries you want to load
- the report settings

4. Two other tabs, located in the Infrastructure category, in the Part Infrastructure workbench, also interfere with Knowledgeware applications.

- General
- Display

5. Change these options according to your needs.

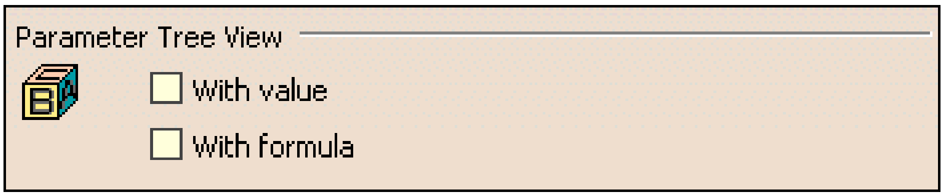
6. Click **OK** when done to validate your settings.

Knowledge

This page deals with these categories of options:

- Parameter Tree View
- Parameter names
- Relations update in part context
- Design Tables

Parameter Tree View



There are 2 types of items that you can display in the specification tree.

With value

Displays the parameter values in the specification tree.

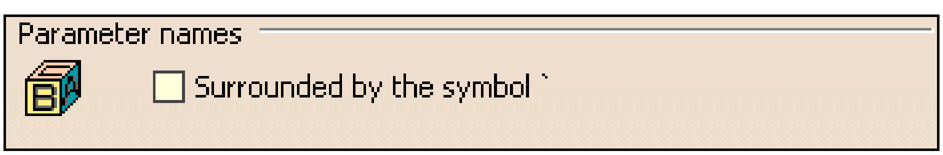
By default, this option is unchecked.

With formula

Displays the formulas constraining the parameter in the specification tree.

By default, this option is unchecked.

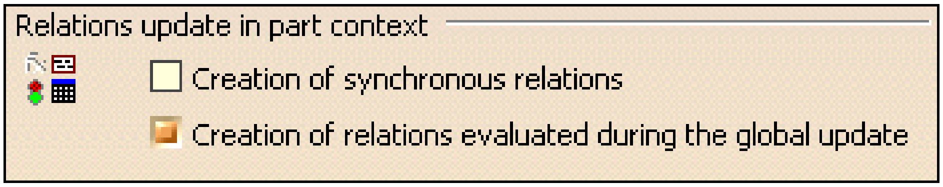
Parameter names



This option should be checked if you work with non-Latin characters. If this option is unchecked, parameter names should have to be renamed in Latin characters when used in formulas.

By default, this option is unchecked.

Relations update in part context



Before V5R12, Knowledge relations (formulas, rules, checks, design tables, and sets of equations) used to execute as soon as one of their inputs was modified.
The user can now choose, when creating the relation, if it will be synchronous (i.e. the evaluation will be launched as soon as one of its parameters is modified) or asynchronous (i.e. the evaluation will be launched when the Part is updated). Each relation can therefore be synchronous or asynchronous.

The 2 following options enable the user to create synchronous or asynchronous relations.

Creation of synchronous relations

Enables the user to create synchronous relations, that is to say relations that will be immediately updated if one of their parameters/inputs is modified. Relations based on parameters are the only one that can be synchronous.

By default, this option is unchecked

Creation of relations evaluated during the global update

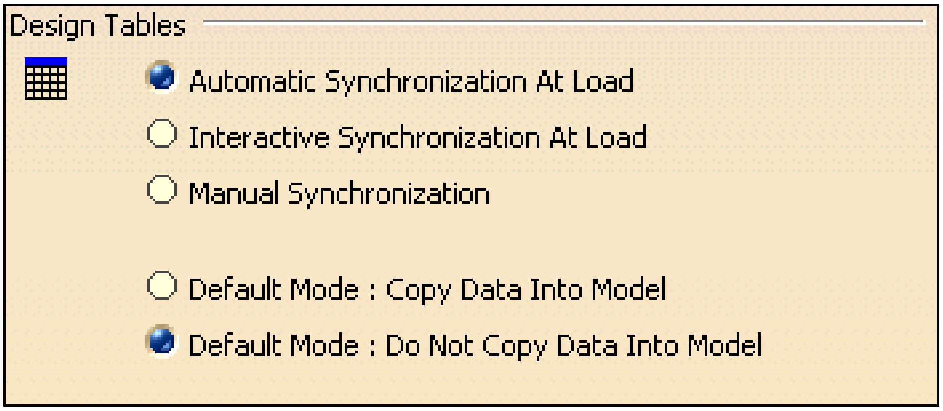
Enables the user to associate the evaluation of asynchronous relations with the global update. The relations can be asynchronous for 2 reasons:

- The user wants the relations to be asynchronous
- The relation contains measures.
- Relations based on parameters: These relations can be synchronous or asynchronous.
- Relations based on geometry: These relations can only be asynchronous.
- Relations based on parameters and on geometry: For the part of the relations containing parameters, the user decides if he wants the update to be synchronous or not. For the other part of the relations, the update occurs when the global update is launched.

Note that the user can also decide if already existing relations are synchronous or asynchronous. To know more, see Controlling Relations Update in the Infrastructure User's Guide.

By default, this option is checked.

Design Tables



There are 2 types of items that you can set up.

Automatic Synchronization at Load

When loading a model containing user design tables, if the design table files have been modified and the external file data is contained in the model, the design table will be synchronized automatically if this radio button is checked.

By default, this option is checked.

Interactive Synchronization at Load

When loading a model containing user design tables whose external source file was deleted, this option enables the user to select a new source file or to save the data contained in the design tables in a new file.

By default, this option is unchecked.

Manual Synchronization

When loading a model containing user design tables, if the design table files have been modified and the external file data is contained in the model, the design table will be synchronized if this radio button is checked. To synchronize both files, right-click the design table in the specification tree and select the **DesignTable object->Synchronize** command or the **Edit->Links** command.

By default, this option is unchecked.

Default Mode: Copy Data Into Model

If checked, the data contained in the external source file will be copied into the model.

By default, this option is unchecked.

Default Mode: Do Not Copy Data Into Model

If checked, the data contained in the external source file will not be copied into the model.

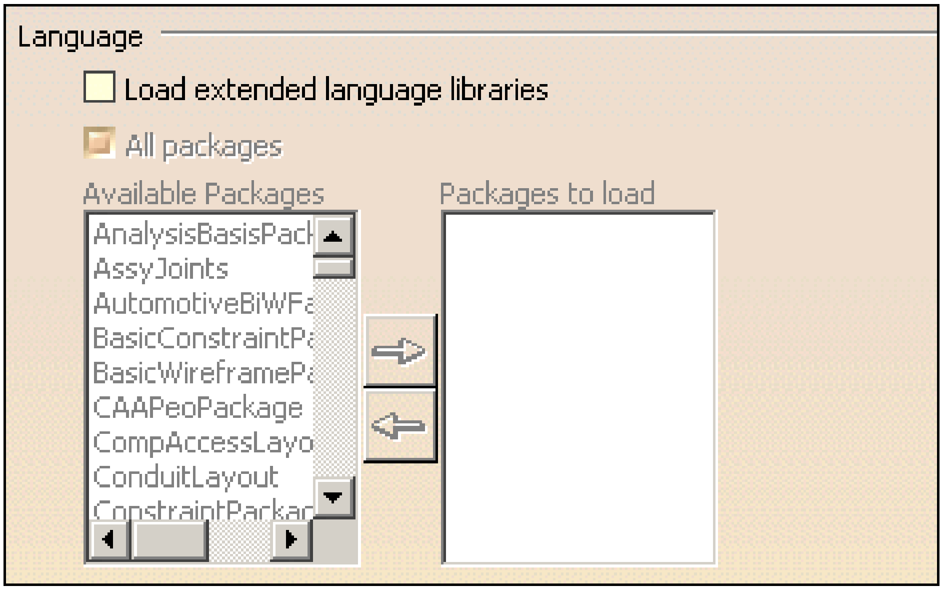
By default, this option is checked.

Language

This page deals with the following categories of options:

- Language
- Reference Directory for Types

Language



This field is to be used when using *measures* in relations or *user functions*. Measures are specific functions to be used in formulas and rules.

The *Knowledge Advisor User's Guide* provides you with tasks explaining how to use measures. For how to create and use user functions, see the *CATIA* Application Architecture documentation.

Load extended language libraries

If checked, enables the user to select the packages he wants to load under Packages (if he wants to load a limited number of packages.)

This option is particularly useful for the administrator to limit the number of packages used by the user. It is also very useful to improve performances since only the required libraries are loaded.

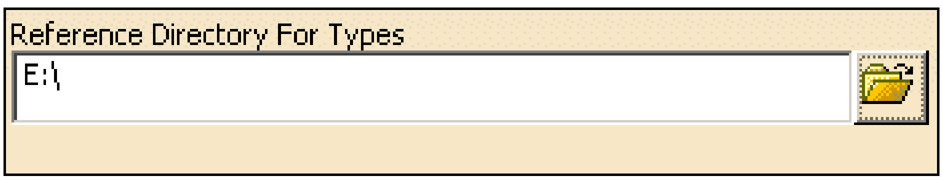
- When you open a document and some relations are broken, you might need to load all libraries to solve the error, which may take quite a long time.
- It is strongly recommended to identify the packages you will need and to select them.

By default, this option is unchecked.

All packages

Enables the user to select all packages.

Reference Directory For Types



Reference Directory For Types

Enables the user to save the CATGScript file in the Directory indicated in the Reference Directory for Types field for later re-user (To know more, see the PKT User's Guide).

By default, this option is not available.

Report Generation

This page explains how to customize the reports generated by the Global Check Analysis tool in the Knowledge Advisor and Knowledge Expert workbenches. It deals with the following categories of options:

- Configuration of the Check Report
- Input XSL
- Report content
- Output directory
- HTML Options

Configuration of the Check Report

Html

Enables the user to generate a HTML report.

Xml

Enables the user to generate a XML report.

By default, the HTML option is enabled.

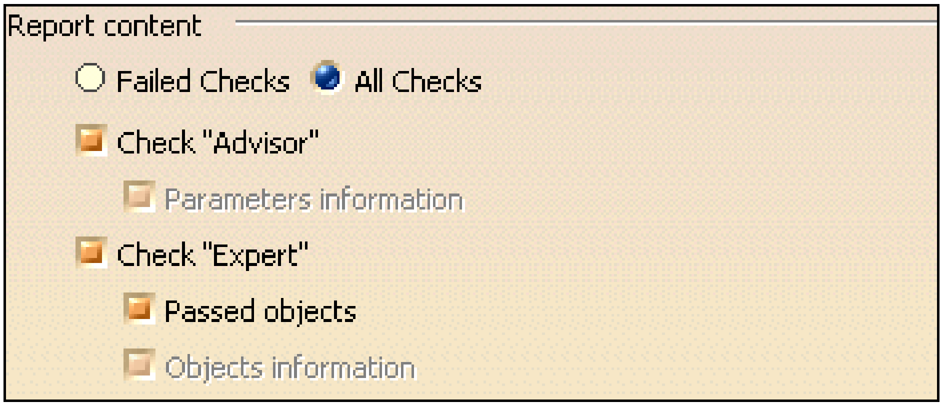
Input XSL

Note that this option is available only if the XML configuration setting is set.

Input XSL

Enables the user to select the XSL style sheet that will be applied to the generated XML report. The StyleSheet.xsl file is the default XSL file, but you can use your own template.

Report content



Failed Checks

If checked, the generated report will contain information about the failed checks only.

By default, this option is unchecked.

All Checks

If checked, the generated report will contain information about all the checks contained in the document.

By default, this option is checked.

Check Advisor

If checked, the generated report will contain information about all the Knowledge Advisor checks contained in the document.

By default, this option is checked.

Parameters information

Not available

Check Expert

If checked, the generated report will contain information about all the Knowledge Expert checks contained in the document.

By default, this option is checked.

Passed objects

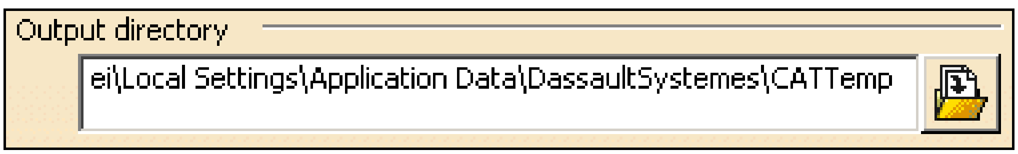
If checked, the generated report will contain information about the objects that passed the Expert checks as well as information about the parameters of these objects (diameter, depth, pitch,...).

By default, this option is checked.

Objects information

Not available

Output directory



Output directory

Enables the user to select the output directory that will contain the generated report.

By default, this option is available.

HTML Options

Open HTML browser into CATIA Session

This option is available for Windows only. It enables the user to define if the report will be opened in a *CATIA* session (in this case, the check box should be checked) or if it will be opened in an Internet Explorer session (in this case, the check box should remain unchecked.)

Note that it is highly recommended not to use this report as a basis for macros or for other applications. It is only provided for information purposes.

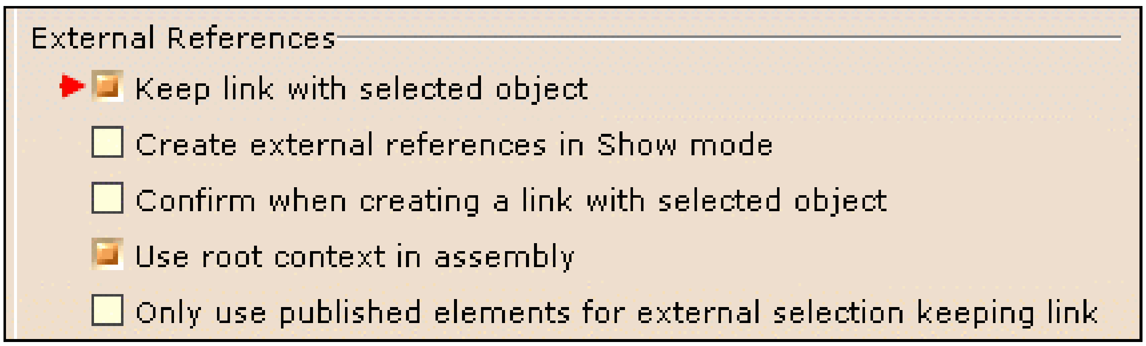
By default, this option is checked.

Part Infrastructure for Knowledgeware Applications

This page deals with the options concerning:

- the external references: Keep link with selected object
- the specification tree display

Part Infrastructure General option

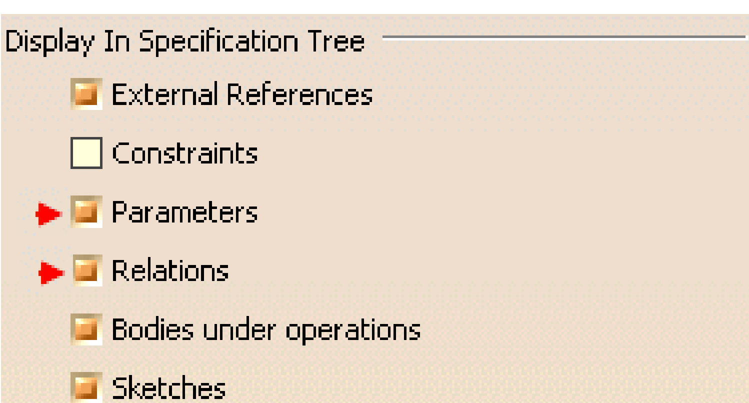


Keep link with selected object

You need to select this option to take advantage of the associativity.

Click here to know more about the Part Infrastructure **General** options.

Part Infrastructure Display option



Parameters

Select this option to display the parameters when working in a CATProduct context.

By default, this option is unchecked.

Relations

Select this option to display the relations when working in a CATProduct context.

By default, this option is unchecked.

Click here to know more about the Part Infrastructure **Display** options.

Glossary

Many of the definitions included in this glossary are only pertinent within the *CATIA* knowledgeware context.

& A C D F K M P W

Symbols

| (operator)	Breaks a single line message into a multiple line message. Can only be used in the Message function when programming rules and checks.
.CATScript	The extension of a macro file generated by *CATIA* Version 5. A macro file can be specified as the argument of the LaunchMacroFromFile function which can be called in rules and checks.
.txt	The extension of a human-readable file composed of text characters. This file format can be used as an import file format when importing parameters and formulas.
.xls	The extension of an Excel file. This file format can be used as an import file format when importing parameters and formulas under Windows.

A

activity	A property which defines whether a relation is applied to a document or not. The activity value is either true or false. It is indicated by an icon in the specification tree and can also be read in the document parameter list.
association	A link between a document parameter and its equivalent parameter in an external design table. Associations are to be created when the document parameter names do not correspond exactly to the parameter names read in the design table.

C

check	A set of statements intended to provide the user with a clue as to whether certain conditions are fulfilled or not. A check does not modify the document it is applied to. A check is a feature. In the document specification tree, it is displayed as a relation that can be activated and deactivated. Like any feature, a check can be manipulated from its contextual menu.

configuration A row in the design table. A configuration is a consistent set of parameter values that can be applied to a document.

D

design table A table containing values to be potentially applied to a document to manage its parameter values. It can be created from the document parameters or from an external file. A design table is a feature. In the document specification tree, it is displayed as a relation that can be activated or deactivated. Like any feature, a design table can be manipulated from its contextual menu.

dictionary The set of parameters, operators, keywords, functions and other components that make up the language to be used to write formulas, rules and checks. The formula, rule and check editors provide you with an interactive view of the dictionary.

F

formula A relation specifying a constraint on a parameter. The formula relation is a one-line statement. Its left part is the parameter to be constrained, the right part is a relation taking as its variables other parameters. A formula is a feature. In the document specification tree, it is displayed as a relation that can be activated or deactivated. Like any feature, a formula can be manipulated from its contextual menu.

K

knowledgeware The set of software components dedicated to the creation and manipulation of knowledge-based information. Knowledge-based information consists of rules and other types of relations whereby designers can save their corporate know-how and reuse it later on to drive their design processes.

Knowledge Inspector An analysis tool which helps users understand how changing any property of their design (such as material, pressure, or a dimensional parameter) changes the operation or design of the product on which they are working. The Knowledge Inspector offers two options:

- "What if" to examine interactions of parameters with each other and with the rules that make up the product's specifications
- "How to" to see how a design can be changed to achieve a desired result

M

magnitude type parameter A parameter whose value is defined by a quantity expressed in specific units. Length, Angle, Time parameters are magnitude type parameters. Boolean, Real, String and Integer parameters are not magnitude type parameters.

P

parameter A feature defining a document property.

R

reaction A Knowledgeware Advisor feature that reacts to events on an object called the source.

relation A knowledgeware feature which, depending on certain conditions:

- sets parameter values
- displays a message
- or runs a macro.

Knowledgeware relations are formulas, checks, rules and design tables.

rule A set of instructions, generally based on conditional statements, whereby the relationship between parameters is controlled. In addition, depending on the context described by the rule instructions, *CATIA* macros can be executed and messages can be displayed. A rule is a feature. In the document specification tree, it is displayed as a relation that can be activated or deactivated. Like any feature, a rule can be manipulated from its contextual menu.

W

wizard A form of user assistance that guides the user through a difficult or complex task within an application. The formula wizard helps the user typing formulas by picking up parameters either in the dictionary, or in the geometry area or in the specification tree.

Index

A B C D E F G H I K L M N O P Q R S T U V W

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W