Product Knowledge Template

Overview

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

This overview provides the following information:

Product Knowledge Template in a Nutshell

Interactive Creation of intelligent Product Knowledge Templates

The encapsulation of feature, part and assembly specifications of any level of complexity within Product Knowledge Template  allows the capture of the design methodology defined interactively in *CATIA*, as well as the reuse across the extended enterprise. Templates can be extracted from existing features, part and assembly designs or created specifically. They can contain not only geometry, but also any associated parameters or relations, including embedded intelligence of design rules, design tables and checks, providing the ability to encapsulate the specifications of sophisticated adaptive features.

Editing and maintaining the Product Knowledge Templates

The management of captured knowledge is easier thanks to the ability to edit and maintain templates in the same interactive way as modifying any *CATIA* part design (no programming skills required). This allows the templates to be adapted to the changing requirements of the organization, to accommodate improved methodologies or customer requirements.

Storing of Product Knowledge Templates in *CATIA* Catalogs for re-use

Once defined Product Knowledge Templates can be stored in *CATIA* Catalogs to allow easy access and management for reuse across the extended enterprise.

Using scripting templates for a powerful solution

CATIA Product Knowledge Templates provides a scripting language enabling full design specification and generation. Using a simple and declarative language, the user can describe - in text format - geometric specifications, transformation and positioning specifications (including assembly constraints), and knowledge specifications.
Moreover, in order to make the creation of scripts still simple, powerful Knowledge Templates can be used as any script objects.
Product Knowledge Template 2 provides the ability to easily and interactively capture engineering know-how and methodology for highly efficient reuse, helping the organization to share best practices and avoid duplication of effort through inability to reuse existing designs.

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 Basic and Advanced Tasks sections.

The Workbench Description section, which describes the PKT 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/pktug/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...
	specific to the P1 configuration
	specific to the P2 configuration
	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

Index

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?

Enhanced Functionality

3D PLM Integration

Document Chooser integration and support of DLNames

You can now customize the document environment in order to select documents or paths using various interfaces (folder, SmarTeam, ENOVIA, and so on). The interface can be customized for a folder or DLName path selection interface. To get an example, see: Instantiating a Part Template From ENOVIA VPM V5 Using the Document Chooser.

Getting Started

Creating a PowerCopy

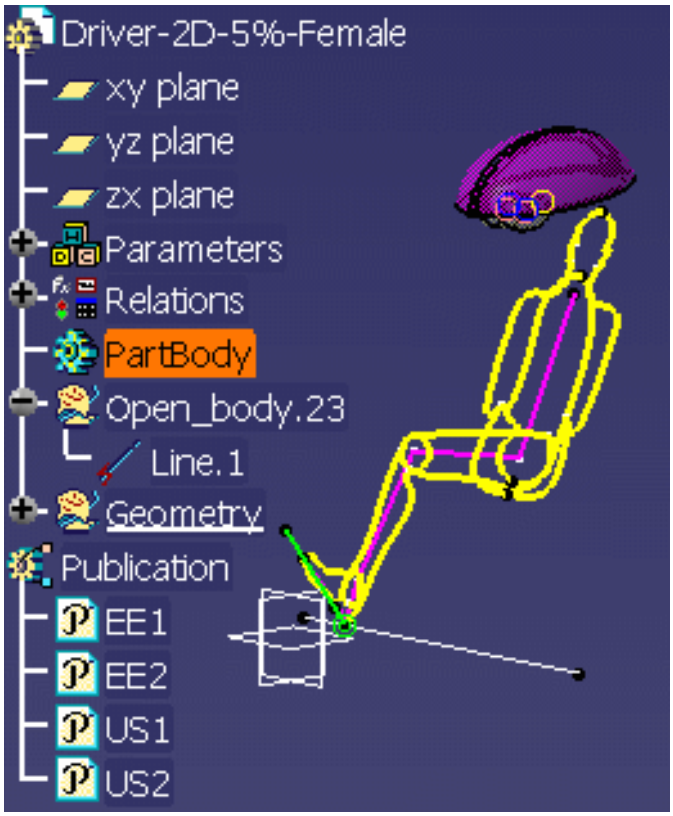
Creating a PowerCopy

This task shows how to create PowerCopy elements, to be reused later.

A PowerCopy is a set of features (geometric elements, formulas, constraints and so forth) that are grouped in order to be used in a different context, and presenting the ability to be completely redefined when pasted.

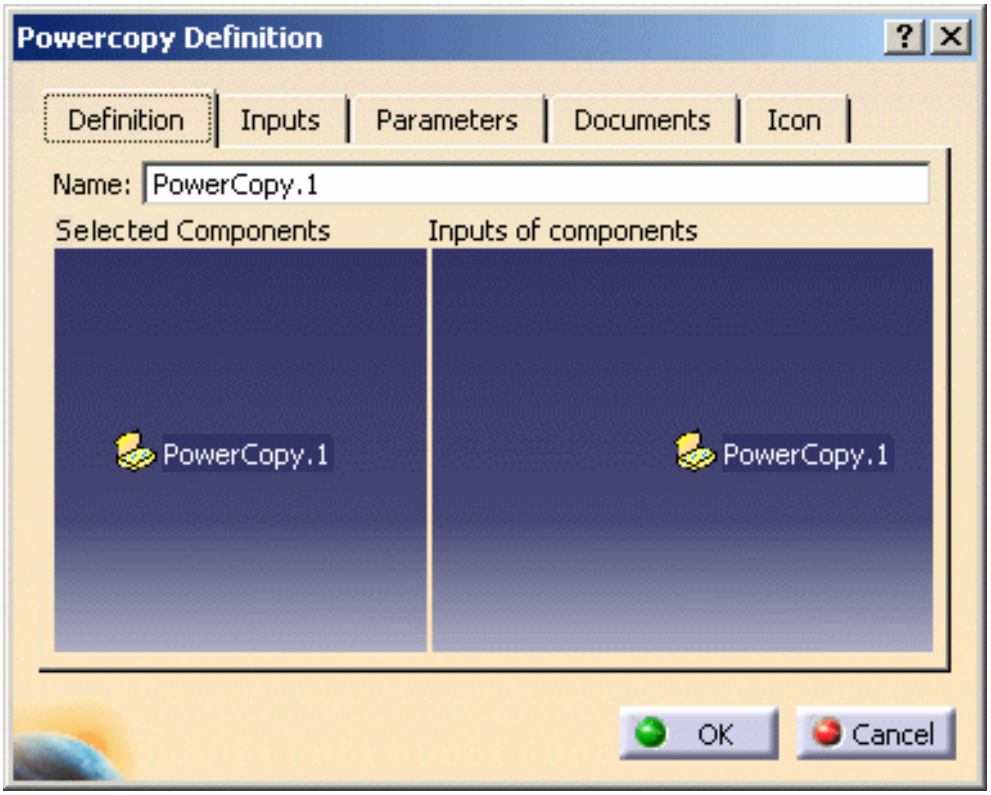
Before carrying out the scenario, make sure that the **Parameters** and **Relations** options are checked (**Tools->Options->Infrastructure->Part Infrastructure->Display**).

1. Open the PktCreatePowerCopy.CATPart document. The following image displays:



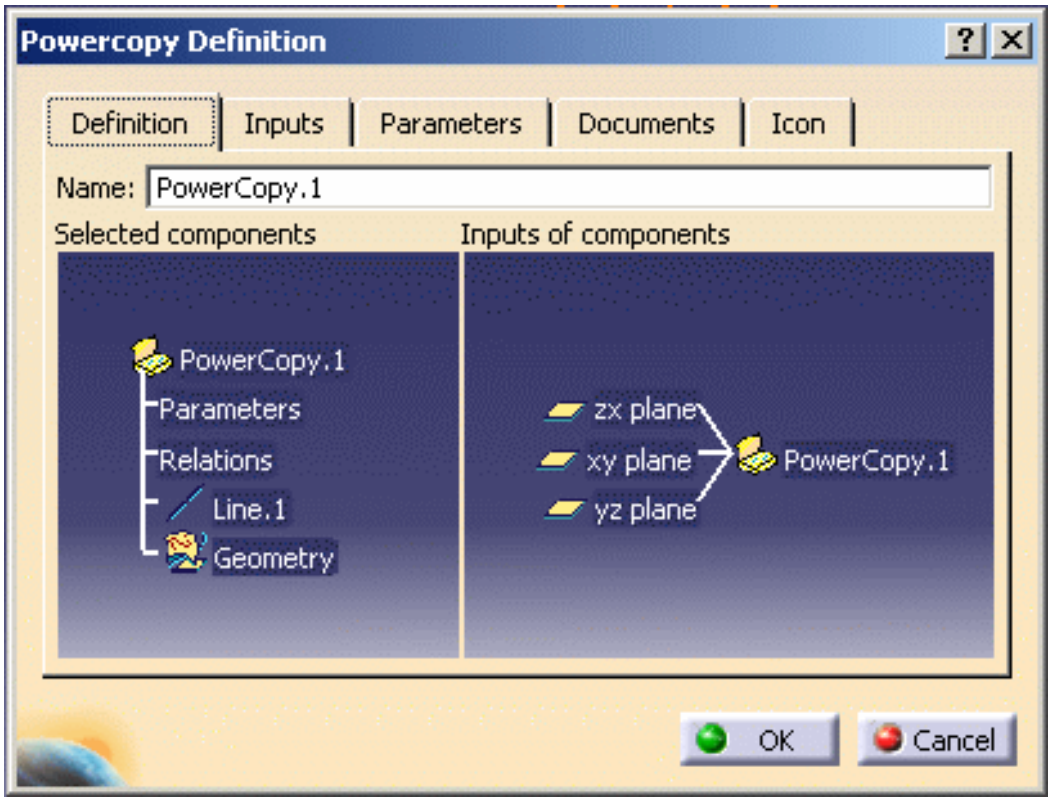
2. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench.

3. Click the **Create a PowerCopy** icon () . The PowerCopy Definition dialog box displays.



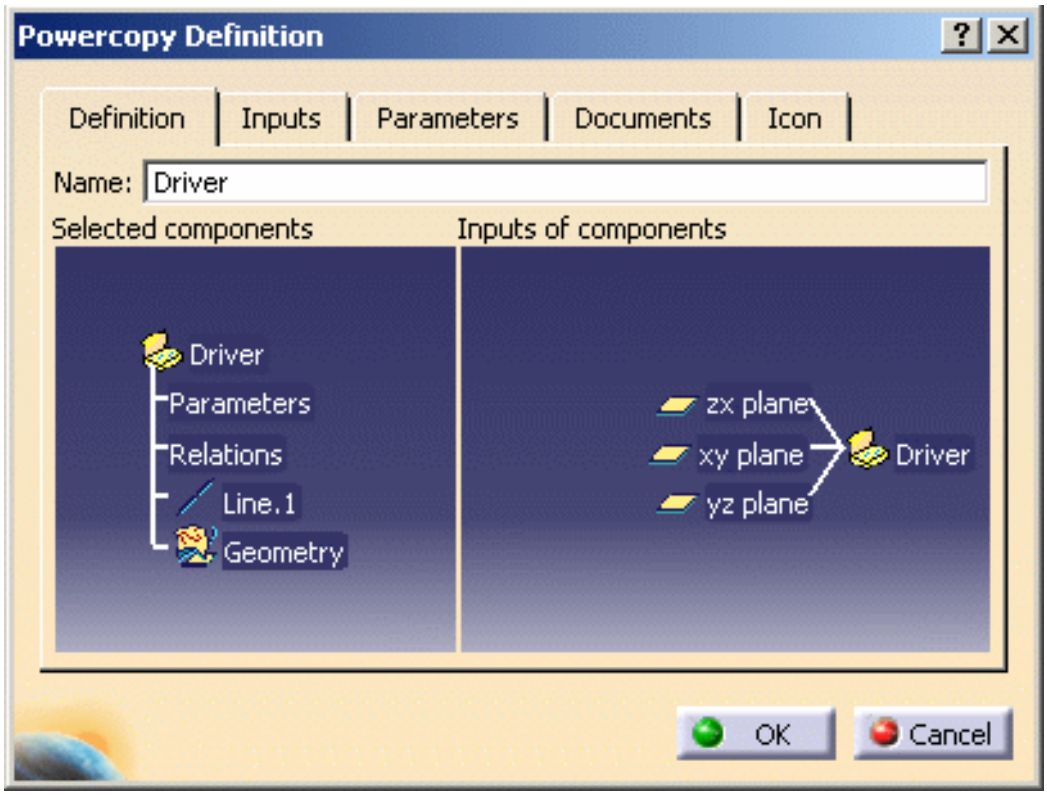
4. Select the elements making up the PowerCopy from the specification tree. For the purposes of our scenario, select the following items:
   - The Parameters
   - The Relations
   - Line.1 (under Geometrical set)
   - The Geometry

The dialog box is automatically filled with information about the selected elements.

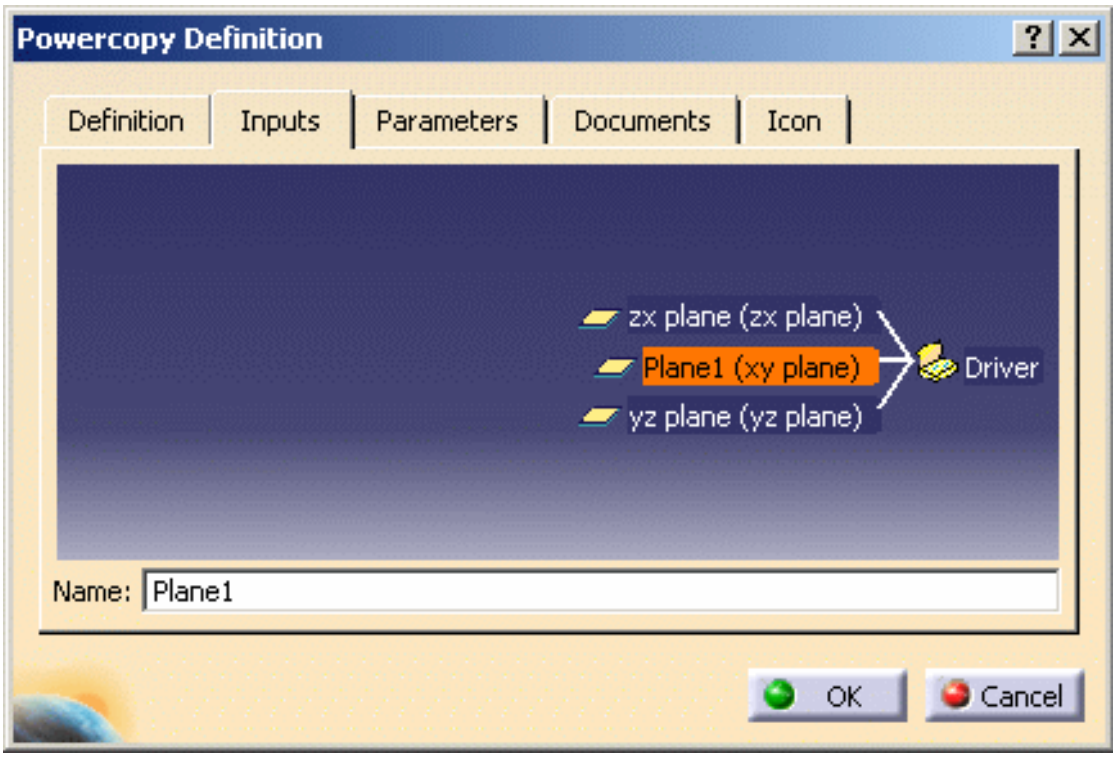


5. Define the PowerCopy as you wish to create it:

- The **Definition** tab lets you assign a name to the PowerCopy and presents its components in the 3D viewer. For example, enter "Driver" in the **Name:** field.



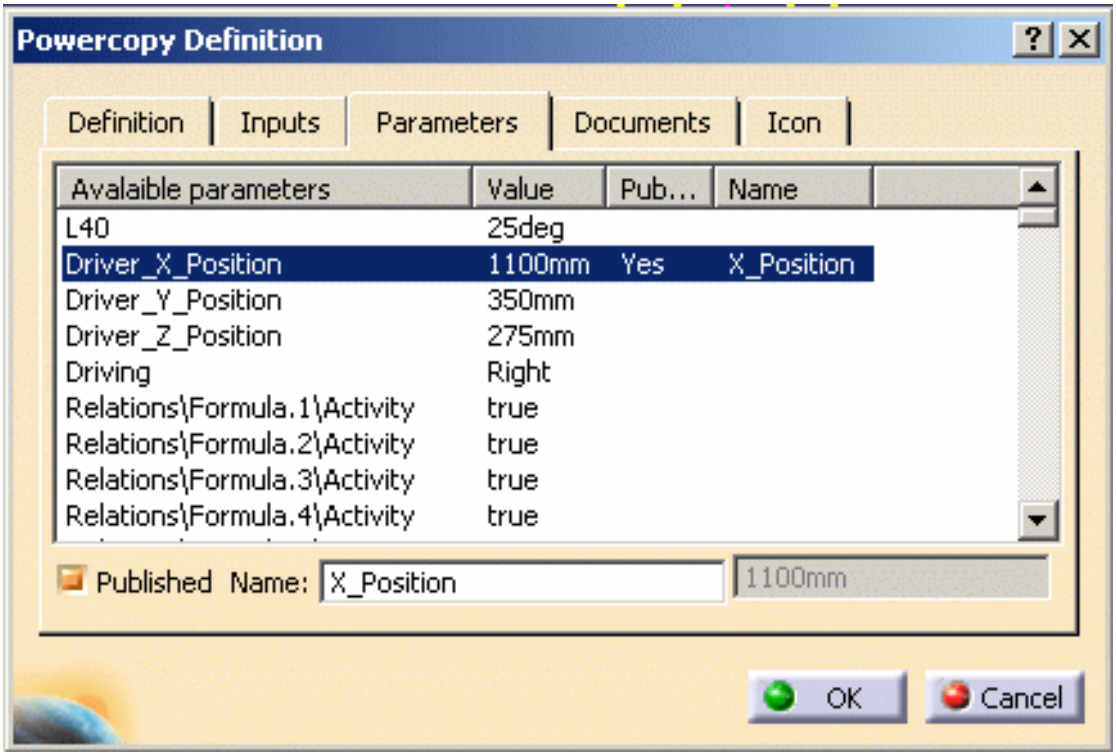
- The **Inputs** tab shows you the inputs (elements to select at instantiation) of the PowerCopy. You can rename these elements for a clearer definition by selecting them in the viewer and entering a new name in the **Name:** field. In parentheses you still can read the elements' default name based on its type. For example, select xy plane and rename it as "Plane1".



- The **Parameters** tab lets you define which of the parameter values used in the PowerCopy you will be able to modify at instantiation time. This can be a value, or a formula for example.

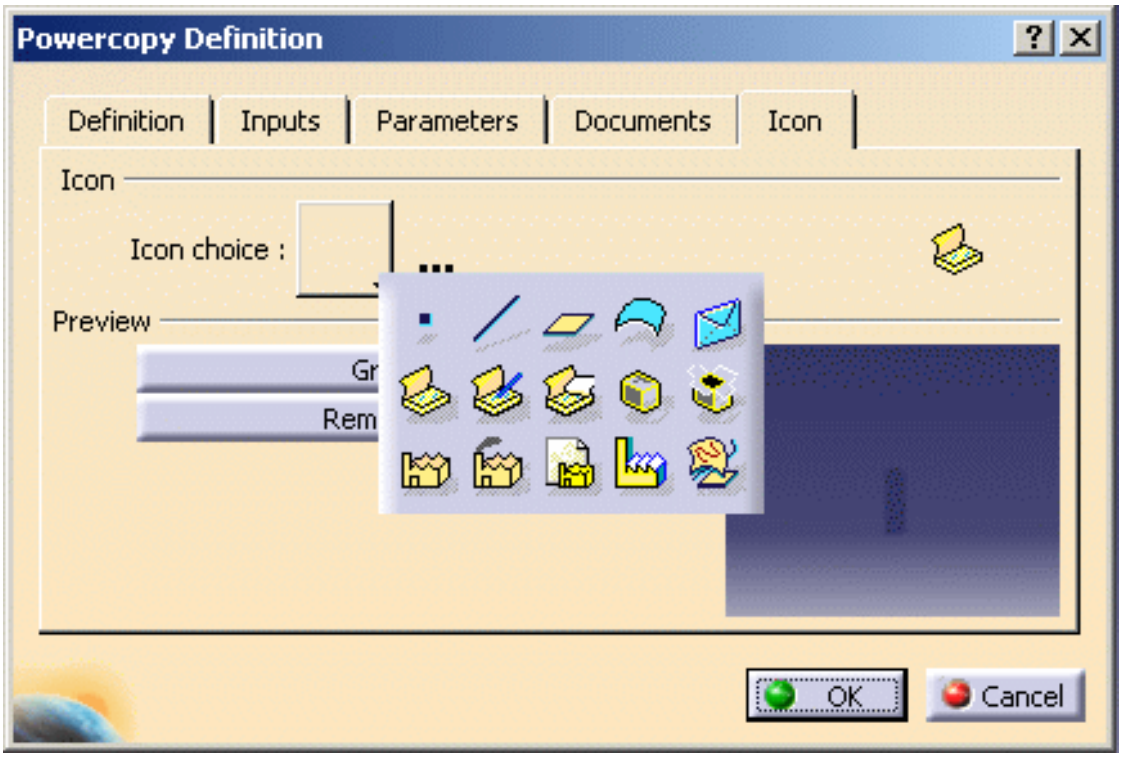
  Simply select the parameters and check the **Published** button. In case of a formula, you can set it to **false** or **true**. For example, select Driver_X_Position.

  Use the **Name** field to give another name to this element. For example, enter X_Position and publish it.



- The **Documents** tab shows the complete path and role of Design tables referenced by an element included in the PowerCopy.

- The **Icon** tab lets you modify the icon identifying the PowerCopy in the specifications tree. A subset of icons is available from the Icon choice button. If you click **...** the Icon Browser opens, showing all icons loaded on your *CATIA* session. Click the envelope icon .



The **Grab screen** button lets you capture an image of the PowerCopy to be stored with its definition. Click the Grab screen button. You can zoom in or out the image to adjust it. Click the **Remove preview** button if you do not need this image.

6. Click **OK** to create the PowerCopy. Save your file. Click here to open the created file.

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on PowerCopies.

Basic Tasks

Managing PowerCopies

Refer to the Quick Reference topic for a comprehensive list of interactions to be carried out on PowerCopies.

Create PowerCopies: Select the **Insert -> Advanced Replication Tools -> PowerCopy Creation** command or click the **Create a Power Copy** icon ( ), select the elements making up the PowerCopy from the specification tree, define a name for the PowerCopy and its reference elements then choose an icon for identifying it.

Instantiate PowerCopies: Select the **Insert -> Instantiate From Document** command or click the Instantiate from Document icon ( ), select the document or catalog containing the PowerCopy, complete the **Inputs** within the dialog box selecting adequate elements in the geometric area or in the specification tree.

Save PowerCopies into a Catalog: Select the PowerCopy from the specification tree, select the **Insert -> Advanced Replication Tools -> PowerCopy Save In Catalog...** command, enter the catalog name and click **Open**.

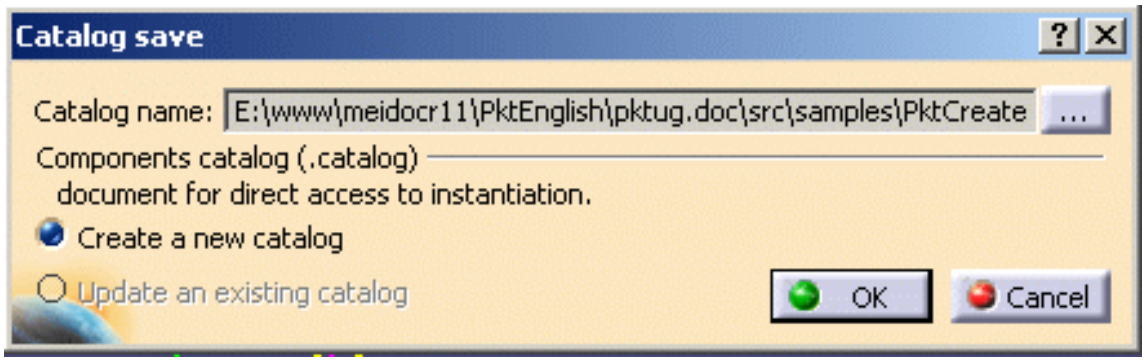
Creating a PowerCopy
Saving a PowerCopy in a Catalog
Instantiating a PowerCopy
Storing a Design Table in a PowerCopy
PowerCopies: Useful Tips
To know more about PowerCopies...

Saving a PowerCopy in a Catalog

This task shows how to store PowerCopy elements into a catalog, for later use as described in Creating a PowerCopy.

1. Open the PktCreatedPowerCopy.CATPart file. The PowerCopy displays below the PowerCopy node.

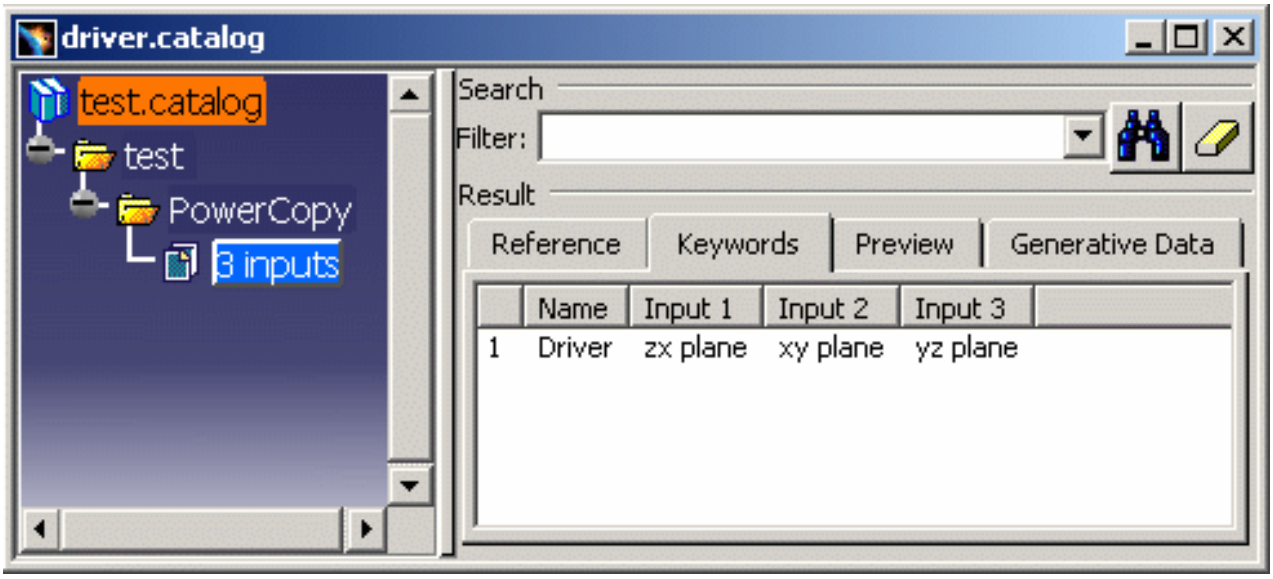
2. Click the **Save in Catalog** icon ( ) from the standard menu bar in the PKT workbench. The 'Catalog save' dialog box displays.



3. Click the **Create a new catalog** option and click the button located on the right-hand side of the **Catalog name** field. The dialog box which is displayed allows you to specify a .catalog file where to store the created PowerCopies. Enter a file name and click **Open**.
4. Click **OK** in the **Catalog save** dialog box.

Note that this command is also available from the Part Design and the GSD workbenches.

5. Open the catalog you have just created (**File->Open** from the standard menu bar). The catalog which is displayed looks like the one below (depending on the name assigned to the catalog):



The left pane displays the PowerCopy created within a tree structure. Selecting 3 inputs displays in the right pane the characteristics of the PowerCopy.

About the Reference tab

The PowerCopy name as well as the document it originates from is displayed.

About the Keywords tab

The PowerCopy name as well as its inputs are displayed.

About the Preview tab

The icon you have associated to the PowerCopy (if any) is displayed.

About the Generative Data tab

The resolved queries: A resolved query is relevant for parts with design tables only since it aims at storing a filtered view of the design table data.

Using the Catalog Editor

1. From the **Start->Infrastructure** menu, access the **Catalog Editor**.
2. Double-click Chapter.1 (default chapter) and click the **Add Family** icon. The Component Definition Family displays.
3. Change the name of the family: Drivers and click **OK**.
4. Double-click the Drivers family and click the **Add Component** icon.
5. In the Description Definition dialog box, click the

button, go back to the

PktCreatePowerCopy.CATPart file and select the Driver PowerCopy in the specification tree and click **OK**.

6. Save your catalog and proceed to the next task: Instantiating a PowerCopy.

To know more about catalogs, see the *Catia Infrastructure User's Guide*.

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on PowerCopies.

Instantiating a PowerCopy

This task shows how to instantiate a PowerCopy once it has been created as described in Creating a PowerCopy:

- from a catalog
- from a document containing a PowerCopy and
- from a selection

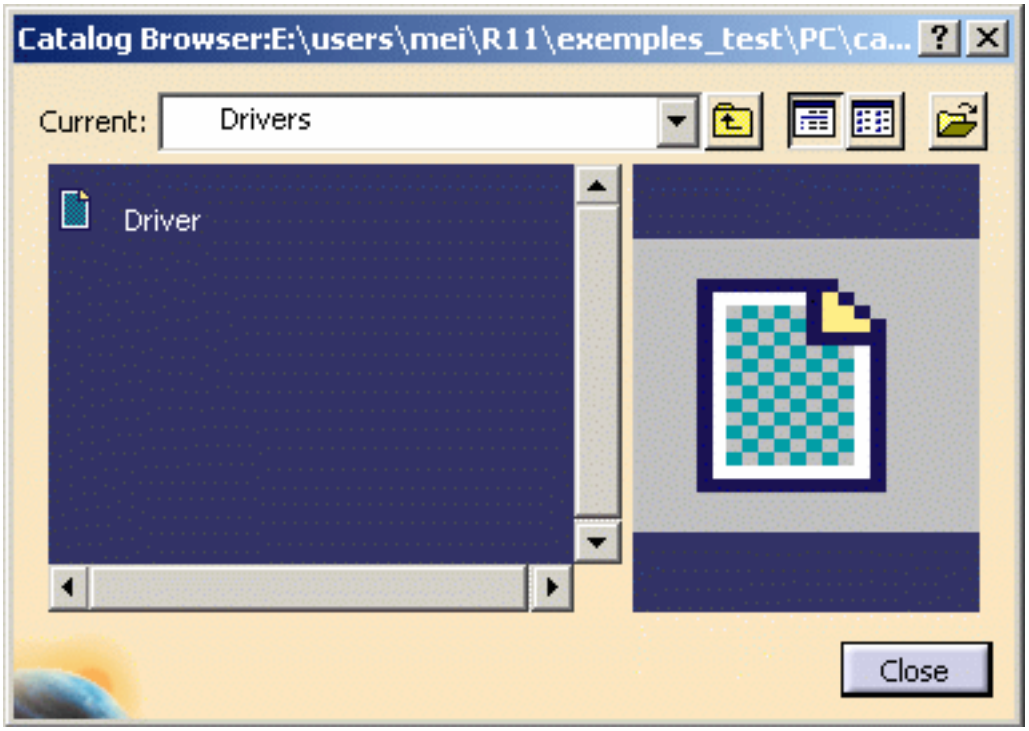
- The PowerCopy reference has links to the selected features. These selected features are the components of the PowerCopy reference.
- Each instantiation of a PowerCopy reference is a copy of each of its components (Copied Features).
- After an instantiation, there is no link between the copied features and the reference components.

From a Catalog

1. Create an empty .CATPart file.
2. In the standard toolbar, click the **Open Catalog** icon (). The catalog browser is displayed. Select the samples/PktPowercopycatalog.catalog file.

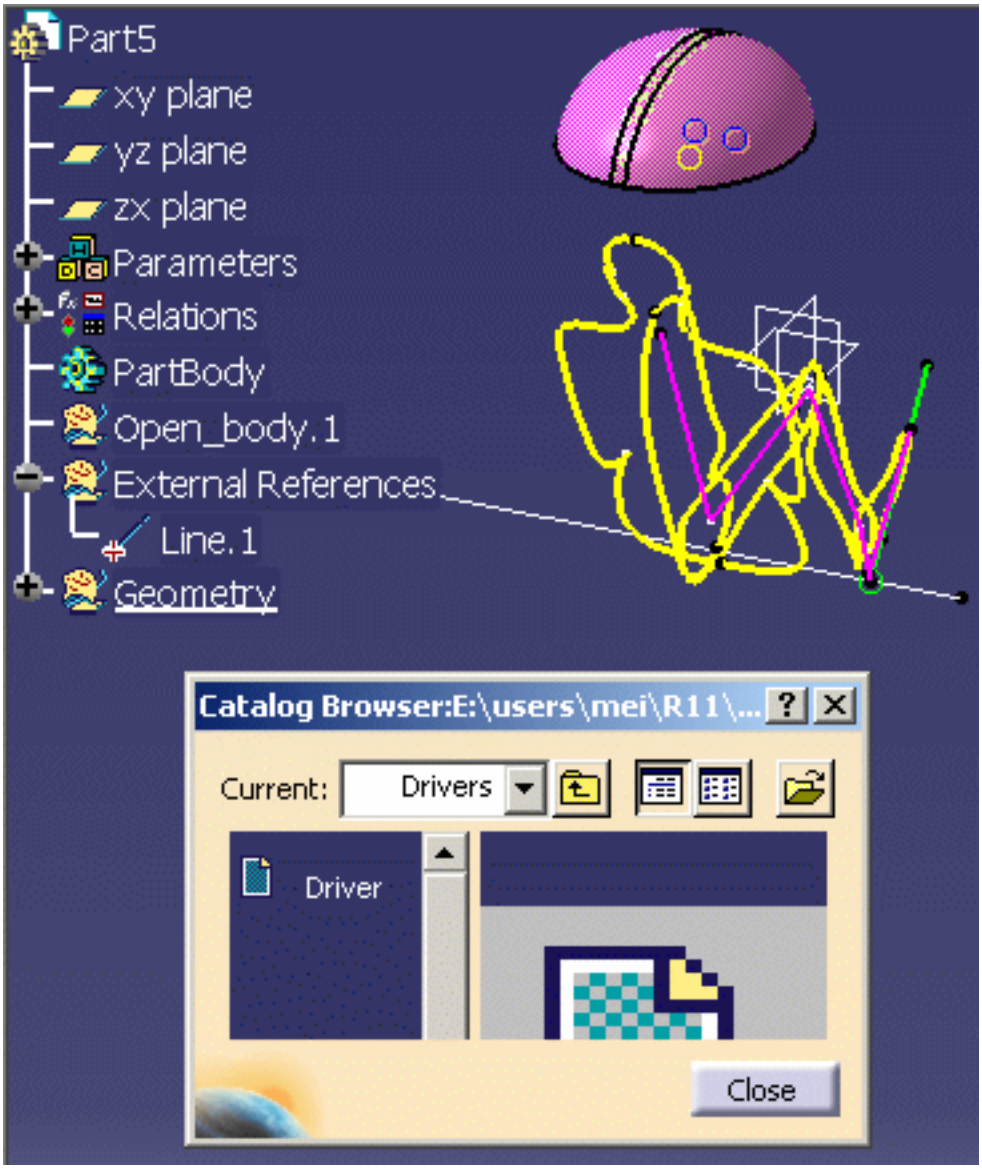
   To know how to store a PowerCopy in a catalog, see Storing a PowerCopy in a Catalog.

3. Click the  icon. In the dialog box which is displayed, select the catalog which contains the PowerCopy that you want to instantiate. Click Open to open the selected catalog. The dialog box which is displayed next enables you to navigate through the chapters and the families of the catalog until you can access the desired PowerCopy.



- To know more about catalogs, see the *Catia Infrastructure User's Guide*.
- To know more about the Insert Object dialog box, click here.

4. Double-click the 'Driver' object. The Insert Object dialog box is displayed.
5. Click the **Use identical name** button and click **OK**. The PowerCopy is instantiated.



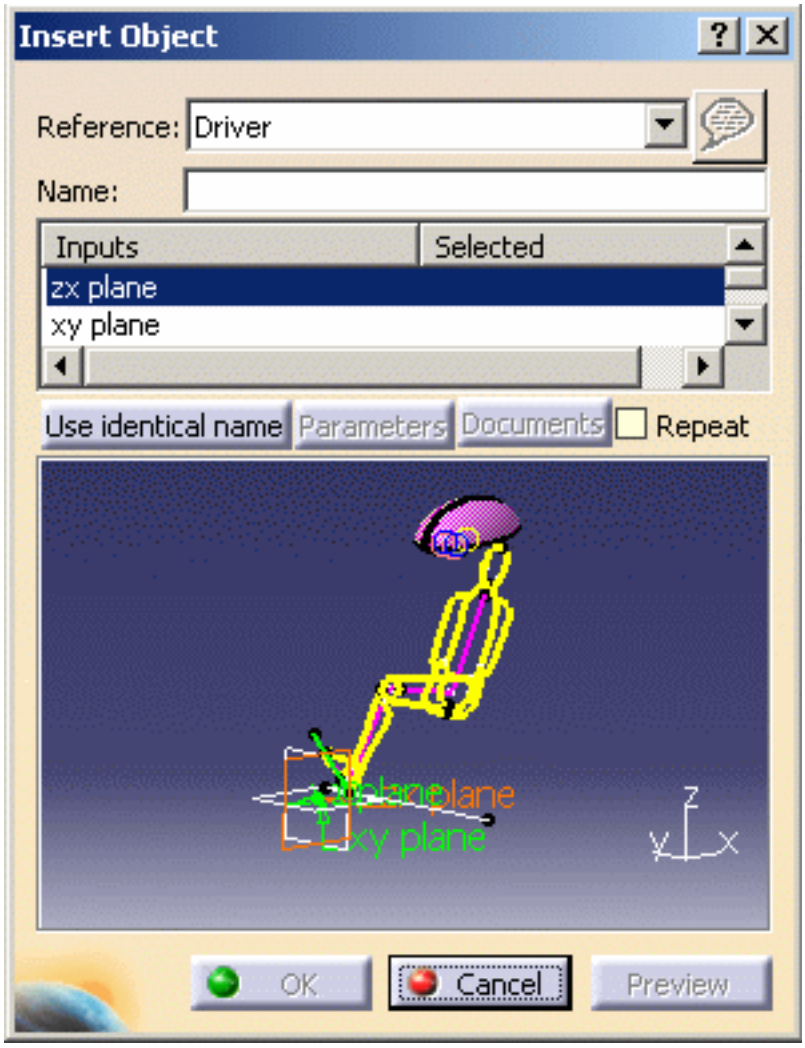
6. Click **Close** in the Catalog Browser when done.

From a Document

1. Open the Pktvehicle.CATPart document.
2. From the **Start->Knowledgeware** Menu, access the **Product Knowledge Template** workbench.
3. Click the **Instantiate from Document** icon ( ). The File Selection dialog box displays.

Note that this command is also available from the Part Design and the GSD workbenches.

4. Select the PktCreatedPowerCopy.CATPart file and click **Open**. The **Insert Object** dialog box displays.

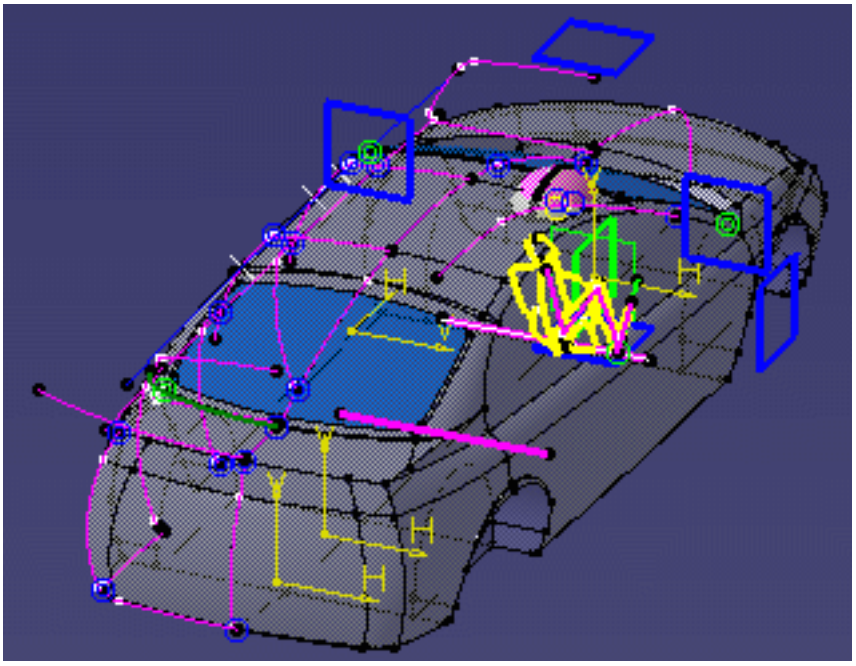


- Click the **Use identical name** button.
- Click **OK** when done.

Click here to know more about the Insert Object Dialog box.

Note that in some cases, when instantiating a powercopy, the replacing element does not present the same sub-elements as the replaced element. Therefore you need to clearly indicate in a specific dialog box, the Replace Viewer, how to rebuild the geometry from the replacing element.

The PowerCopy is instantiated into the document (see picture below.)

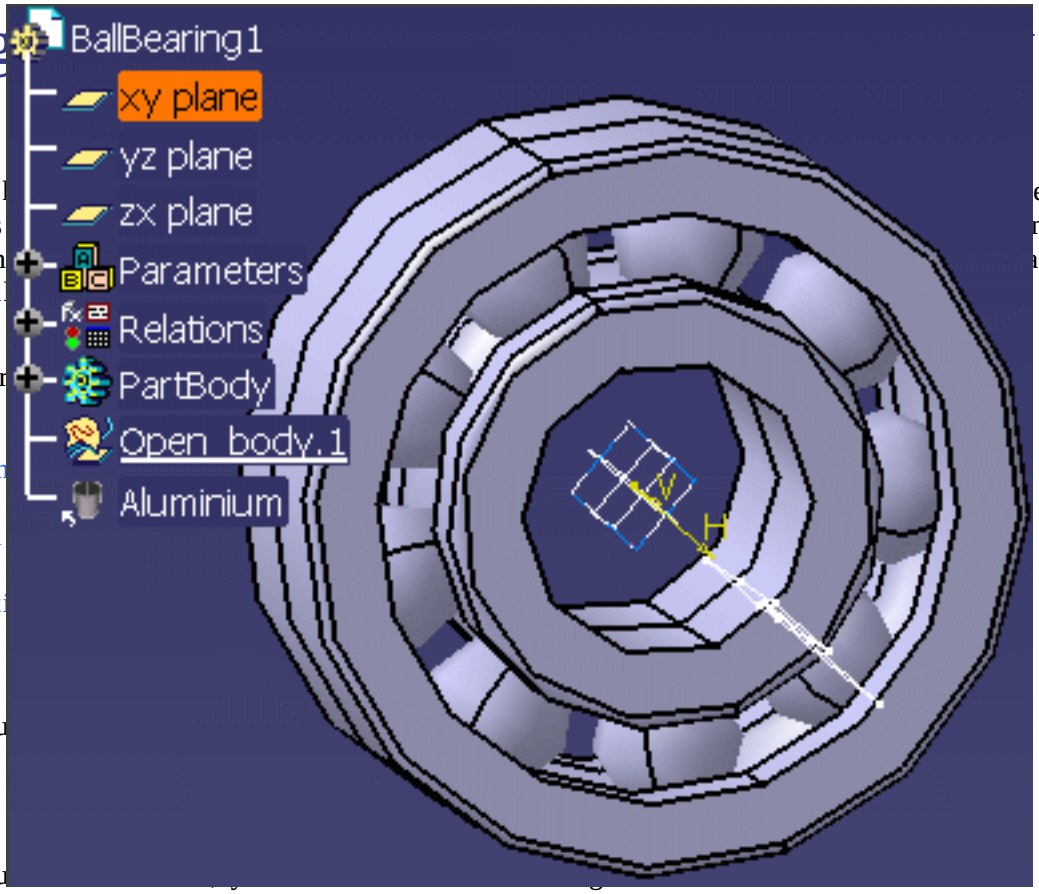


From a Selection

1. Open the PktCreatedPowerCopy.CATPart and the PktVehicle.CATPart files.
2. Tile the window vertically.
3. Expand the PowerCopy node in the PktCreatedPowerCopy.CATPart file and click the Driver PowerCopy.
4. Go to the PktVehicle.CATPart file and click the **Instantiate from Selection** icon ( ). The Insert Object dialog box displays.
5. Click the **Use identical name** button. Click **OK** when done. The PowerCopy is instantiated.

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on PowerCopies.

Storing



This task sh enario, the user wants nt context. To do so, h an already existing ba

This scenar

- Insertin
- Creatin
- Instanti

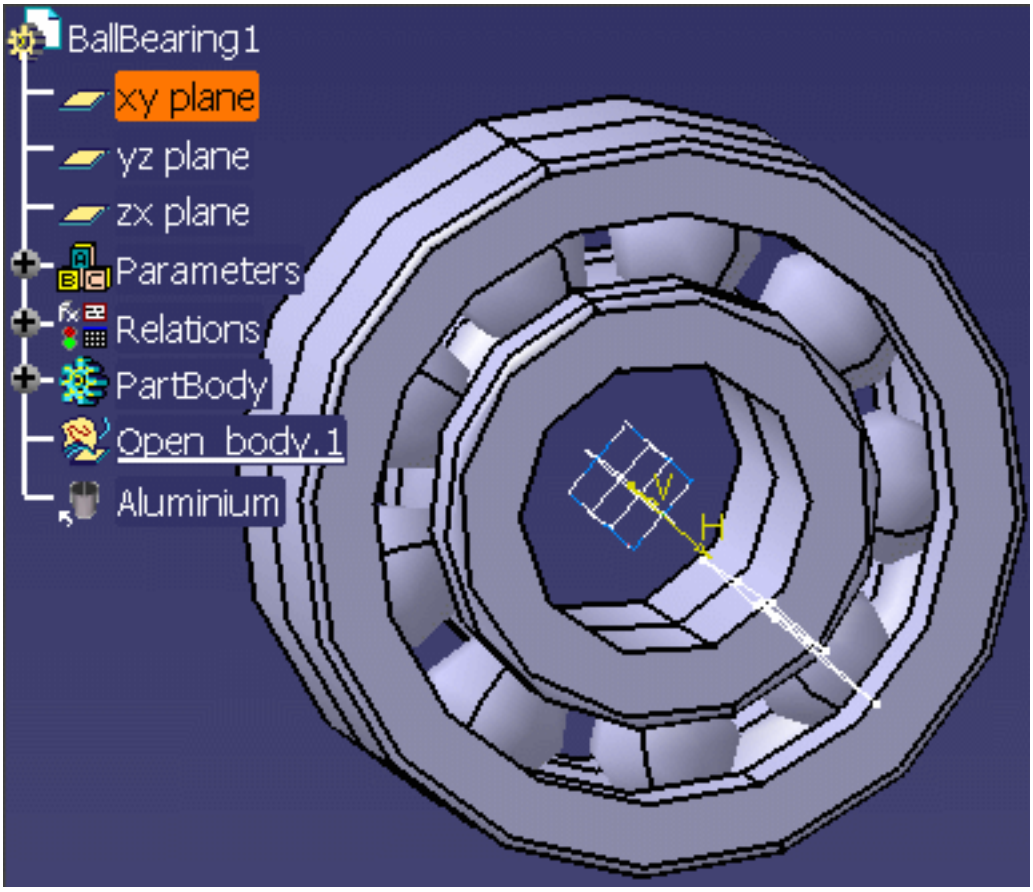
To carry ou

To carry ou

- KwrBallBearing1.CATPart
- KwrBearingDesignTable.xls

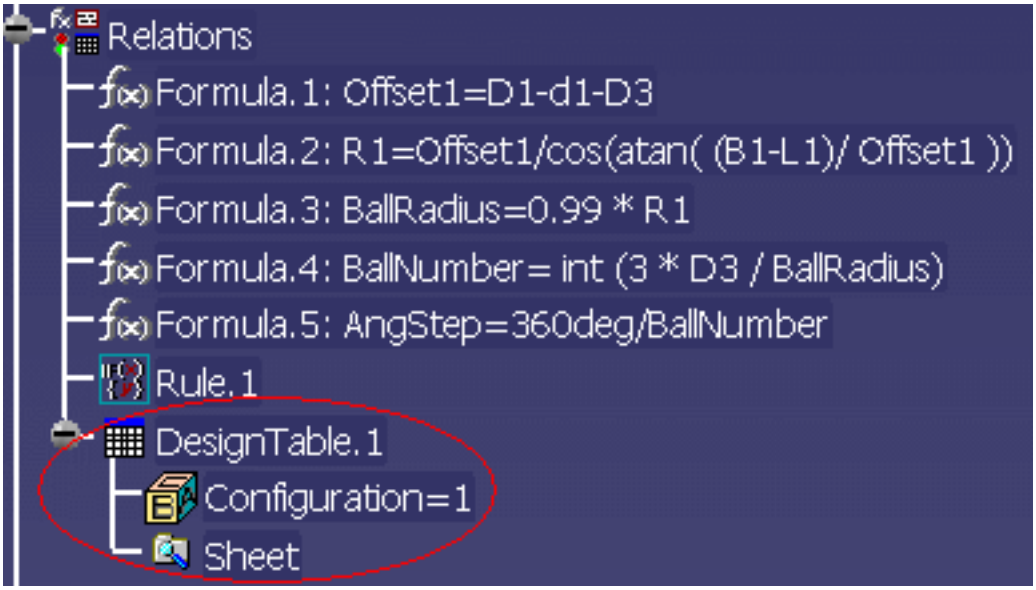
To store a design table in a PowerCopy, do not forget to select the parameters pointed by the design table.

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.



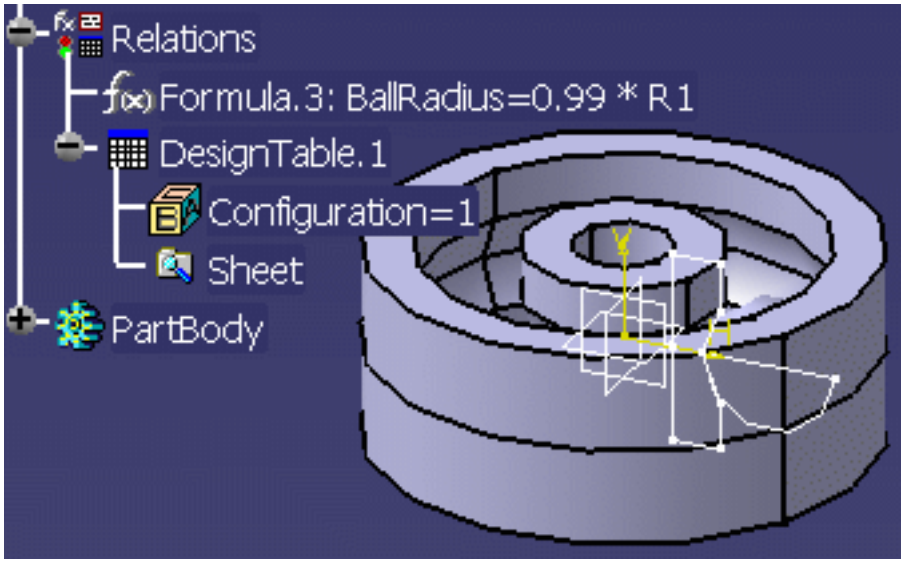
Creating the PowerCopy

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



PowerCopies: Useful Tips

Creating a PowerCopy

- As far as possible, minimize the number of elements making up the PowerCopy.
- When defining PowerCopies including sketches, use profiles constrained with respect to edges or faces rather than to planes. Additionally, set the option Create geometrical constraints off before sketching. Generally speaking, it is always preferable to use profiles both rigid and mobile.
- It is preferable to constrain elements with respect to external references such as faces, edges, reference or explicit planes.
- It is preferable not to use projections nor intersections in your sketch if you want to use your sketch in a PowerCopy.
- Avoid constraints defined with respect to reference planes.
- Before creating your PowerCopies, make sure that your sketch is not over-constrained.
- Make sure that your sketch is iso-constrained (green color). You can use non-iso-constrained sketches, but it will be more difficult to understand and control the result after instantiation.
- Create sketches on an axis system, in order to better control the Sketch position.
- Avoid access to sub-elements.
- Formulas are automatically included if you select all the parameters.
- For complex design, integrate knowledge rules.

Managing inputs:

- Always rename your inputs to help the end user understand what he needs to select.
- A formula is automatically included in a Power Copy definition when all its parameters are included.
  Otherwise, i.e. if at least one parameter is not selected as part of the Power Copy, select the formula to make it part of the definition. If you do so, all the formula parameters that have not been explicitly selected, are considered as inputs of the Power Copy.
- Note that when including parameters sets containing hidden parameters in a PowerCopy, the hidden parameters are automatically instantiated when instantiating the PowerCopy.
- When creating a powercopy, you may select components that point a relation. If this relation is not activated, it will not be taken into account by the powercopy. For this relation to be inserted into the powercopy, you have to activate it.

Preview:

- In a Part document, create only one Power Copy reference. It is not a technical restriction, but there are at least two raisons for this: The cost of an instantiation will be smaller if the Part document is smaller. The end user can more easily understand the feature to be instantiated.
- Put in "show" only the input and the result (to help the end user to understand what he needs to select).
- Use color to differentiate inputs (put transparency on result for example).
- Choose a pertinent viewpoint before saving the Part document reference, default viewpoint in preview during instantiation will be the same.

Catalog:

- Do not forget catalog integration if you want to provide several Power Copies.

Instantiating a PowerCopy

- Always check the orientation for curves and surfaces.
- If you need to instantiate a Power Copy several times on the same input, rename your inputs and use the "Use identical name" option.

To know more about PowerCopies...

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on powercopies.

The **PowerCopy** command can be accessed by selecting the **Insert->Advanced Replications Tools->PowerCopy Creation** command from the following workbenches:

- Part Design
- Generative Shape Design

and by clicking the **Create a PowerCopy** icon () from the Product Knowledge Template workbench.

A PowerCopy is a template that works at the part level. From a collection of features (geometry, literals, formulas, constraints, etc.), the user can create his/her own feature. The result is a Part Design feature or a Shape Design feature that can be reused in the design of another part. The created feature can be saved in a catalog.

Managing User Features (UDFs)

Refer to the Quick Reference topic for a comprehensive list of interactions to be carried out on user features. Refer to To know more about User Features.

Create User Features: Select the **Insert -> UserFeature -> UserFeature Creation ...** command or click the **Create a UserFeature** icon, select the elements making up the User Feature from the specification tree, define a name for the User Feature and its reference elements then choose an icon for identifying it.

Instantiate User Features from Document: Select the **Insert -> Instantiate From Document** command, select the document or catalog containing the User Feature, complete the **Inputs** within the dialog box selecting adequate elements in the geometric area or from the specification tree.

Save User Features into a Catalog: Select the user feature from the specification tree, select the **Insert -> Save in Catalog** command, enter the catalog name and click **Open**.

Introducing the UserFeature Definition window
Creating a User Feature
Creating and Instantiating a NLS User Feature (UDF)
Saving a User Feature into a Catalog
Instantiating a User Feature
Assigning a Type to a User Feature
Referencing User Features in Search Operations
User Features: Useful Tips
User Features: Limitations
To know more about User Features ...

Introducing the User Feature Definition Window

The **Userfeature Definition** window is accessed when selecting the **Insert->UserFeature->UserFeature Creation...** command or when clicking the **Creates a UserFeature** icon ( ).

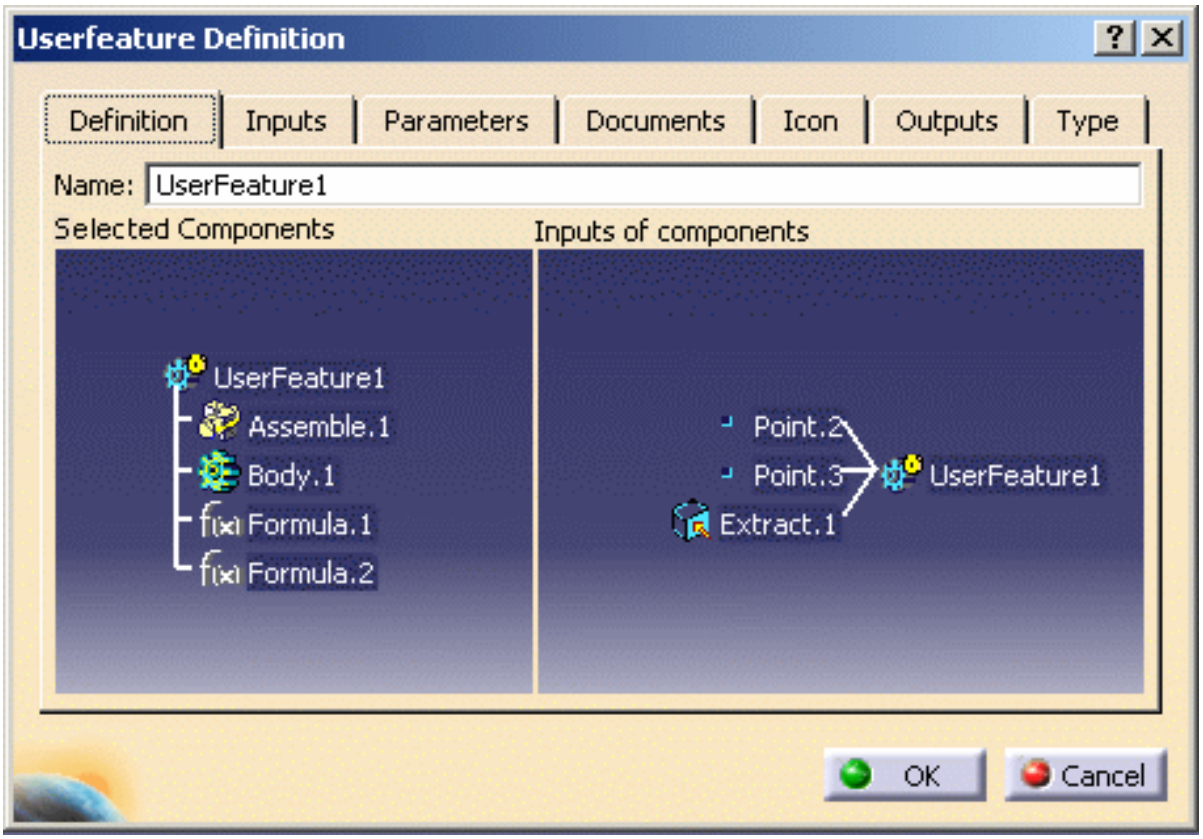
Reference file: PktModifyingMainResult.CATPart

The Definition tab

The Definition tab lets you define the user feature as you wish to create it.

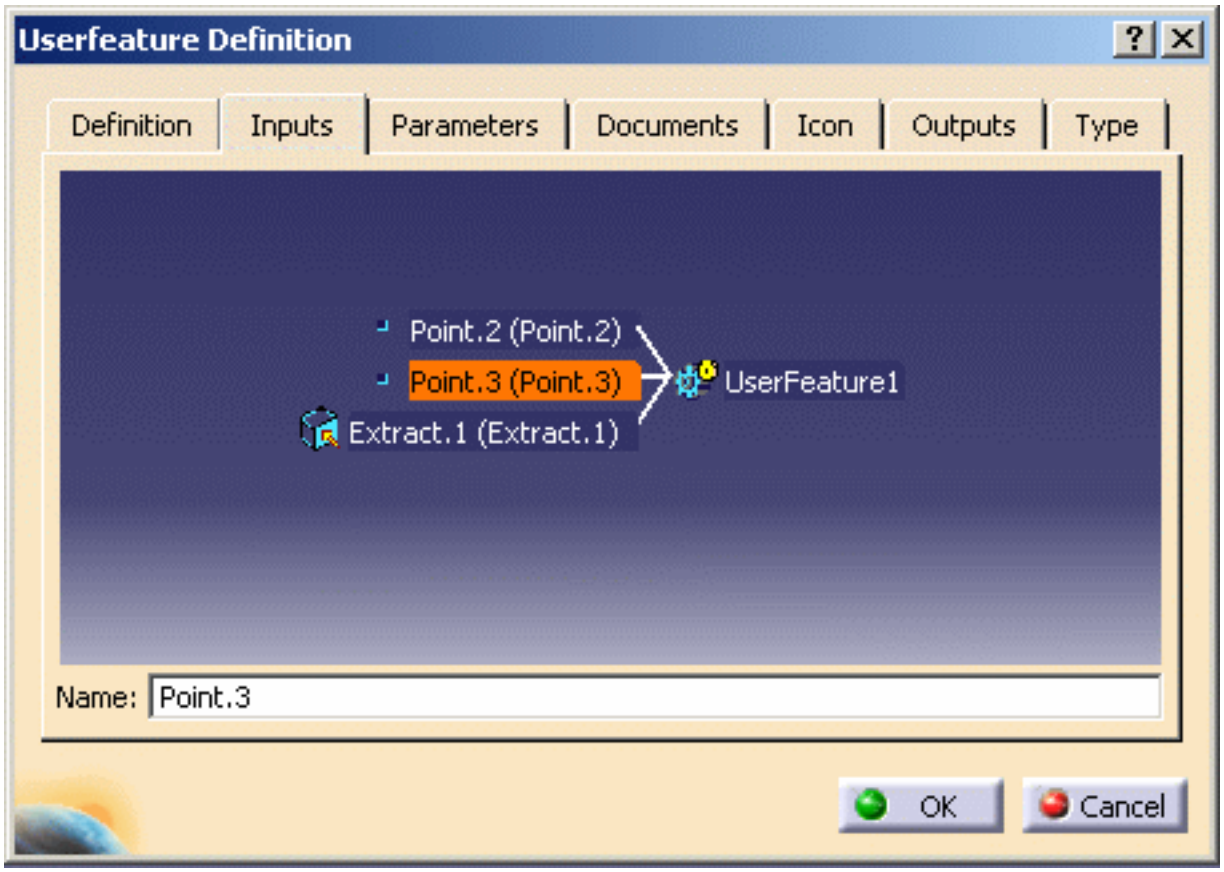
The Definition tab lets you assign a name to the user feature and presents its components in the viewer.

- The tree structure displayed in the Definition tab under **Components** differs from the structure of the selected feature. The Body.1 object does not appear as an Assemble.1 child.
- The components displayed under **Inputs of components** are the features which are not aggregated to the selected object but are required to build it. These are the inputs that will be requested at instantiation.



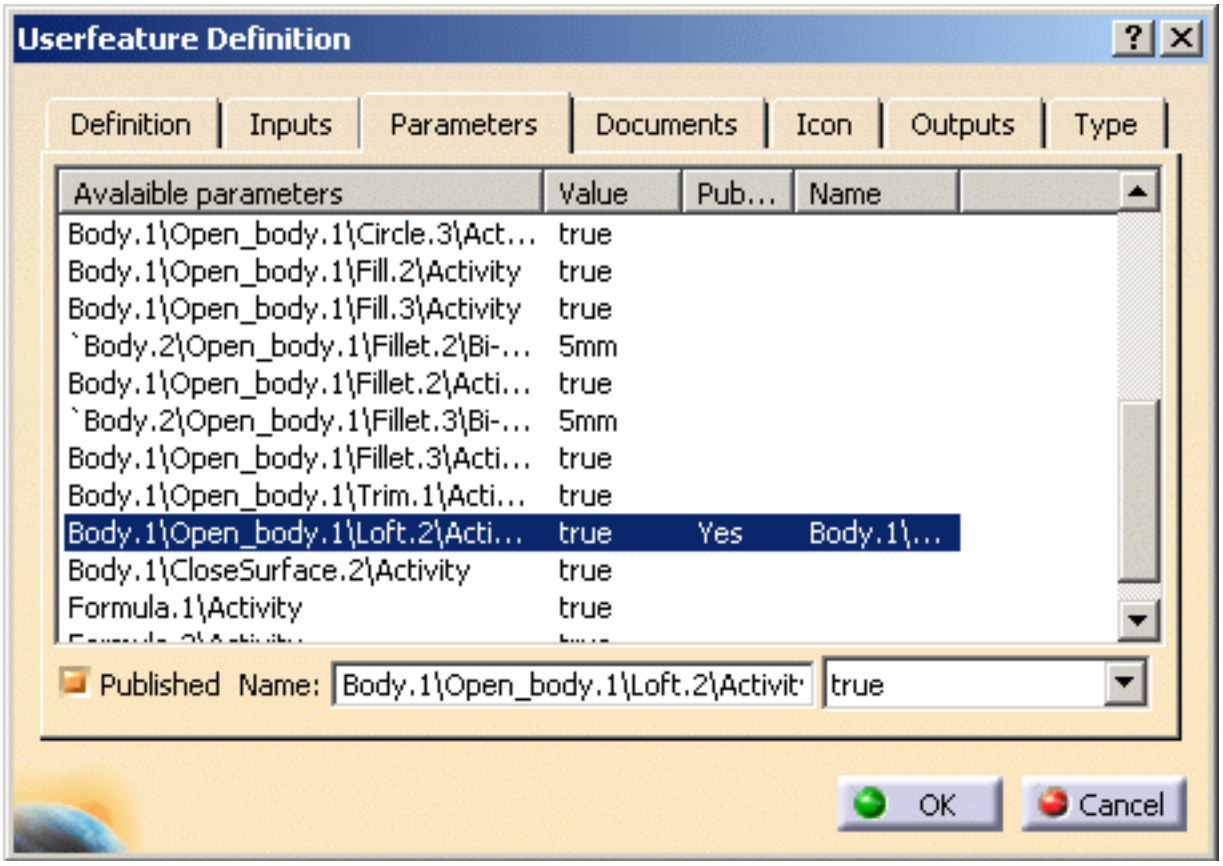
The Inputs tab

- The Inputs tab shows you the inputs (elements to be selected at instantiation) of the user feature. You can rename these elements for a clearer definition by selecting them in the viewer and entering a new name in the Name: field. In parentheses you still can read the elements' default name based on its type. . When the Inputs tab is selected, the user feature inputs are indicated by red arrows in the geometry area. To know more, see Renaming an input.



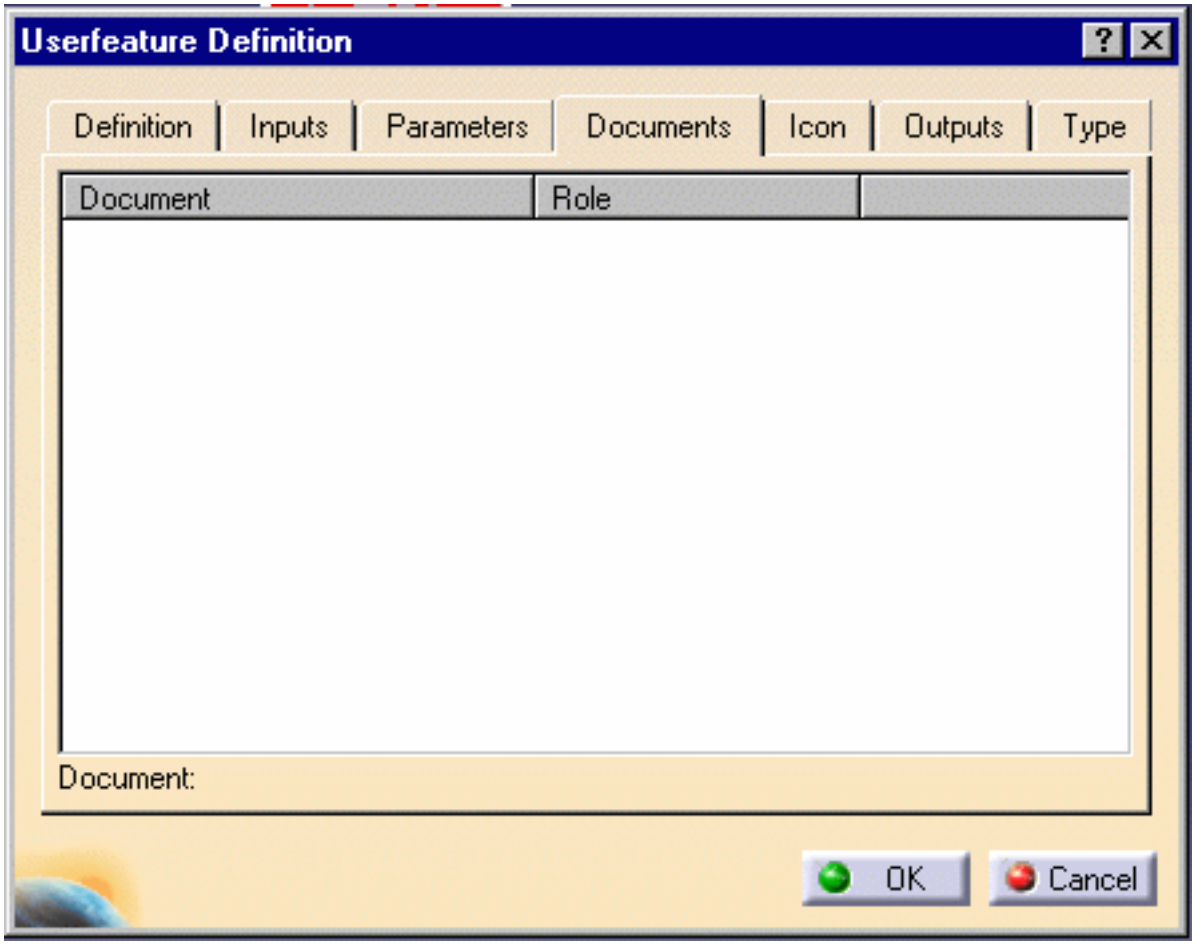
The Parameters tab

- The parameters tab lets you define which of the parameter values used in the user feature you will be able to modify at instantiation. This can be a value or a formula.
  Simply select the parameters and check the Published button. Use the Name: field to give another name to this element. To publish parameters, see Publishing parameters.



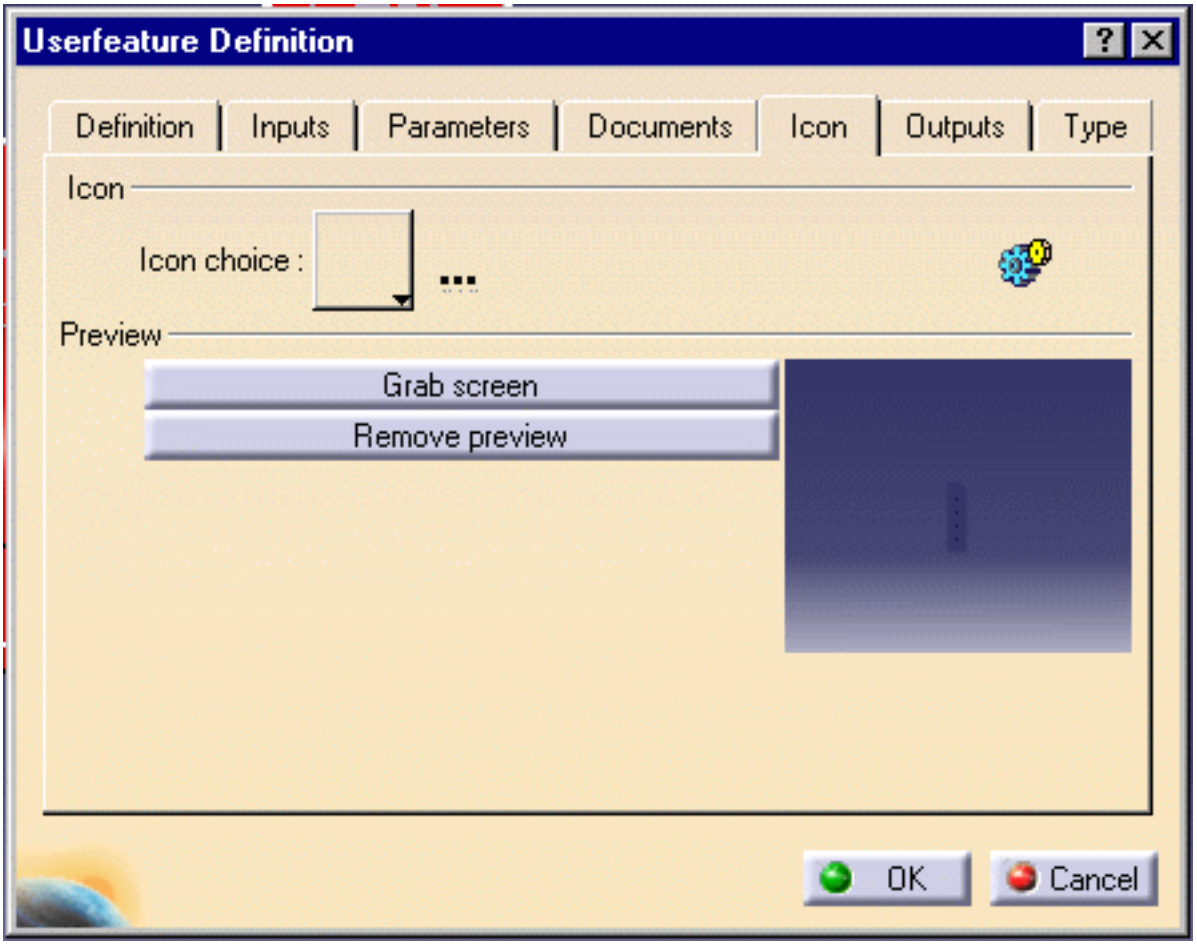
The Documents tab

- The Documents tab shows the complete path and role of Design tables referenced by an element included in the user feature. This tab exhibits no document because only design tables belonging to the selected object are displayed. When instantiating or editing the user feature, you will be able to change the document pointed by the internal design table.



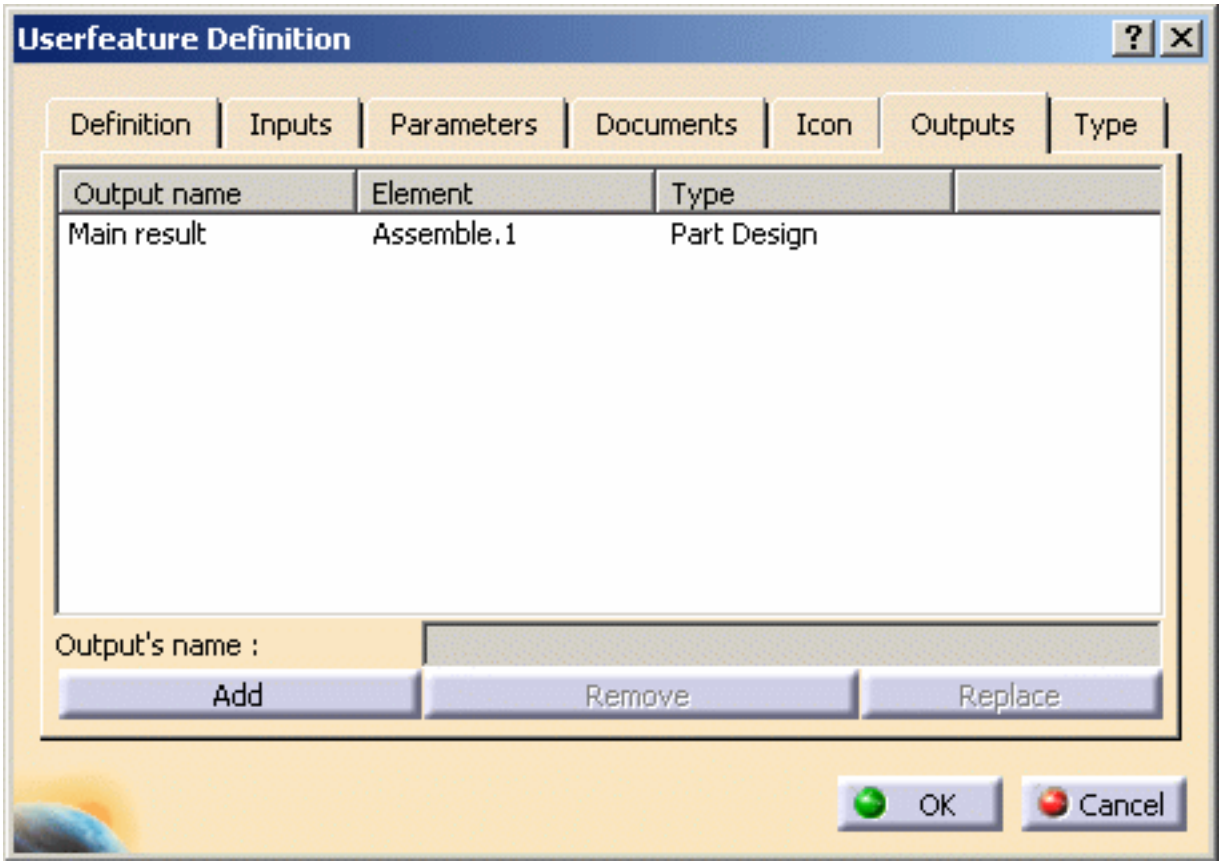
The Icon tab

- The icon tab lets you modify the icon identifying the user feature in the specifications tree. A subset of icons is available from the Icon choice button. If you click ... the Icon Browser opens, showing all icons loaded on your *CATIA* session.
  The Grab screen button allows you to capture an image of the user feature to be stored with its definition. Click the Grab screen button. You can zoom in or out the image to adjust it. Click the Remove preview button if you do not need this image.



The Outputs tab

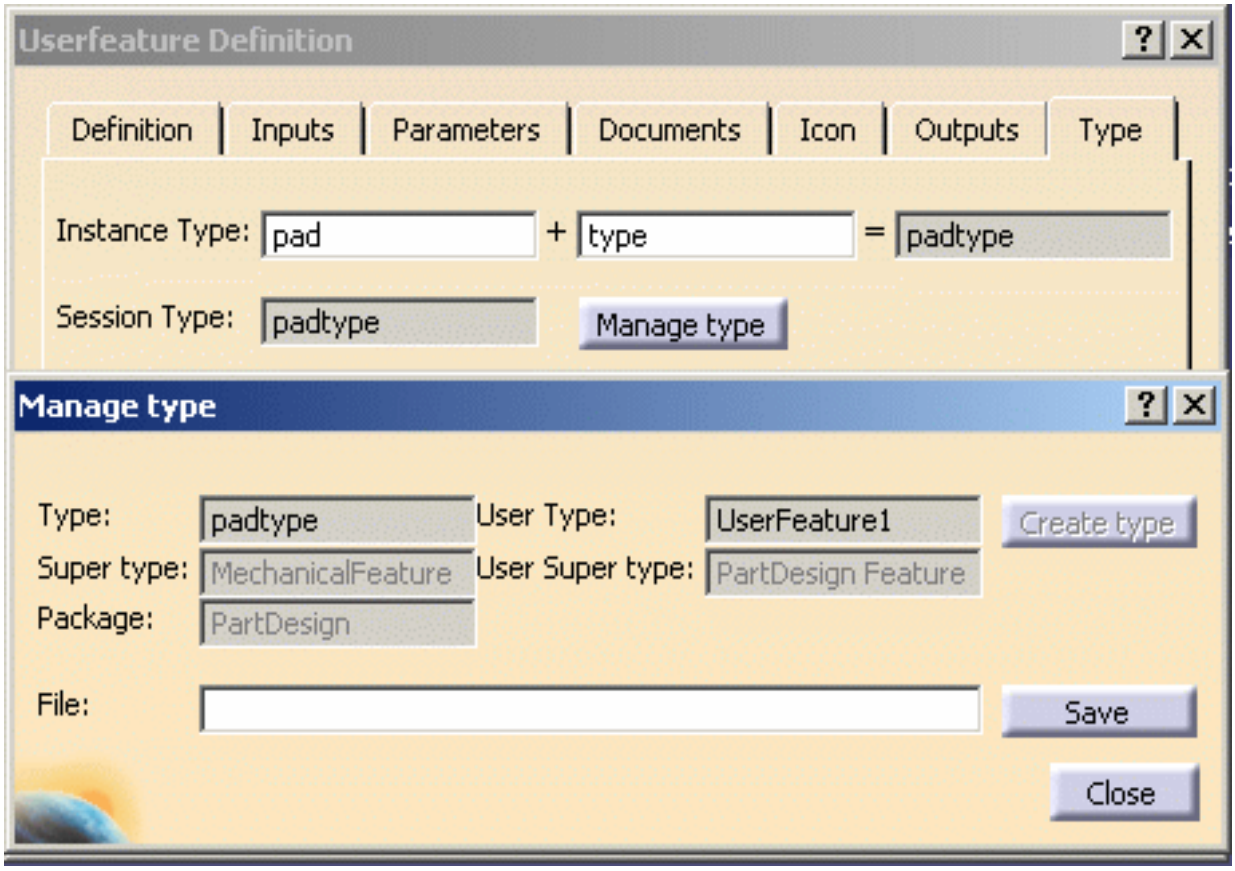
- The Outputs tab provides you with a way to define the result to be carried forward from the user feature to another document during the instantiation process. To know more, see Modifying the Main Result.
  Note that the dimension of the secondary outputs should always be inferior to the Main result.



The Type tab

- The Type tab provides you with a way to associate a type to a user feature. You will be able to use this type in search operations, Expert checks, and in Product Knowledge Template.

The User Type name is the one that you indicated in the **Definition** tab.



Instance Type:

Enables you to define the name of the type.
The first box should contain a string of at least 3 characters and should be used to enter an identifier/keyword.
It is highly recommended not to use special characters in those fields (they are not supported.)
The string to be entered in the second box is not limited.
The third box contains the generated Instance Type name that will be see as the type.

Manage Type button

Enables you to access the Manage Type window where you can:
- Select the Super Type from which the type you created will inherit. It is possible to select the MechanicalModeler, the GSD and the Part Design packages.
- Select the package it will belong to.
- Click **Create type** and **Close** if you want to use the created type in the current session only.
- Click **Create type**, **Save**, and **Close** if you want to use the created type in another session. In this case, a Generative Script file (File field) containing the user feature definition is created.

If you want to reuse the generated type in another Catia session, proceed as follows:

- save the CATGScript file in the Directory indicated in the Reference Directory for Types field (see **Tools->Options->Parameters and Measure->Language** tab)
- check the **Load extended language libraries** check box and select the package containing the type you created.

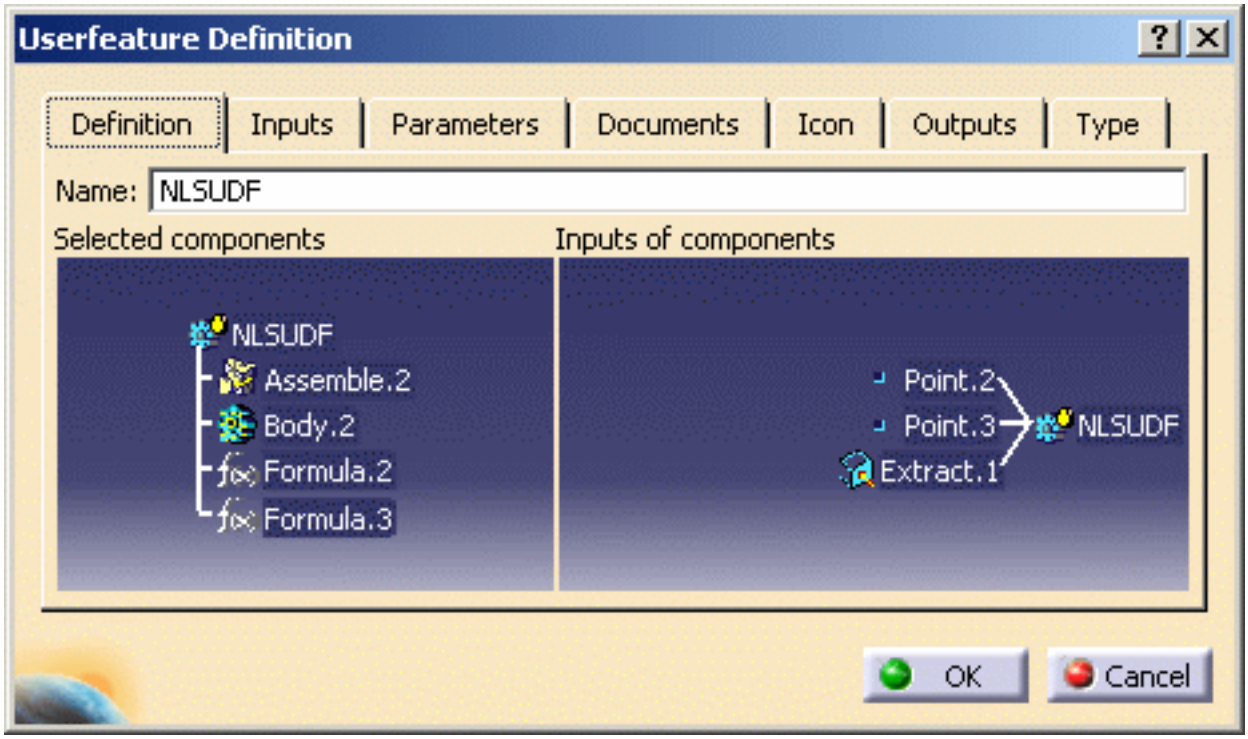
Creating a User Feature (UDF)

The scenario below describes in detail how to create a user feature. A first user feature has already been created. A new user feature is now created.

Note that datums (features that cannot be calculated) cannot be inputs of user features. To know more about the UDF limitations, click here.

1. Open the PktcreateaUDF.CATPart file. Note that this file already contains a UDF located below the KnowledgeTemplates node.
2. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench.
3. Click the **Create a UserFeature** icon ( ). The UserFeature Definition dialog box is displayed.

   Replace the default user feature name with Pad2, then select the Assemble.2 object in the specification tree. The dialog box looks like the one below:



4. Select the **Outputs** tab. By default, the Assemble.2 object is displayed as the main result.
5. Click **OK** in the dialog box. The Pad2 user feature is added to the specification tree.
6. Save your file.
7. Keep this document open and proceed to Saving a User Feature in a Catalog.

To know more about the user feature definition window, see Introducing the Userfeature Definition window. Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on user features.

Creating and Instantiating a NLS User Feature (UDF)

The scenario below describes how to create a NLS user feature. The following features can now display in the user's language at instantiation:

- The input role
- The parameters (names and string multiple values)
- The output role after the instantiation
- The update error message

In the scenario below, the input role and the parameters are NLS.

To create a NLS user feature, proceed as follows:

- Create the user feature and click the Type tab in the user feature definition window to create the type associated to the user feature as well as the associated .CATGScript file.
- Create the CATNls file (See below).
- Close *CATIA* and relaunch it.
- In *CATIA*, open the file into which the template will be inserted.
- Instantiate the user feature.

The created CATNls file:

- Should be stored in the run-time view in the resources\msgcatalog directory.
- Should have the following name: CATType*TypeName*.CATNls. If the type name is Wheel, the CATNls name will be: CATTypeWheel.CATNls.
- Should be structured as follows:
  - Role1="NlsRole";
  - Role2="NlsRole"; ...
  - Optionally for an NLS error message: UpdateErrorMessage = "Message";

Please find below the example of a .CATGscript file and its corresponding .CATNLS file.

```
GSDPackage isa Package
      {
       CATWheel isa SkinFeature
        {
         NLSName = UserFeature1;
         Fill = 0 , Type : Feature ;
         `Main result` = 0 , Type : Feature
         NLSName :`Main result` ;
         Point = 0 , Type : Feature ;
         Plane = 0 , Type : Feature ;
         Configuration = 0 , Type : String ;
         Distance = 0 , Type : LENGTH ;
         Radius = 0 , Type : LENGTH ;
         }
      }
```

In the .CATGscript opposite, the Point, the Plane, the Configuration, the Distance, and the Radius are the inputs of the user feature. These inputs will be required when instantiating the user feature.

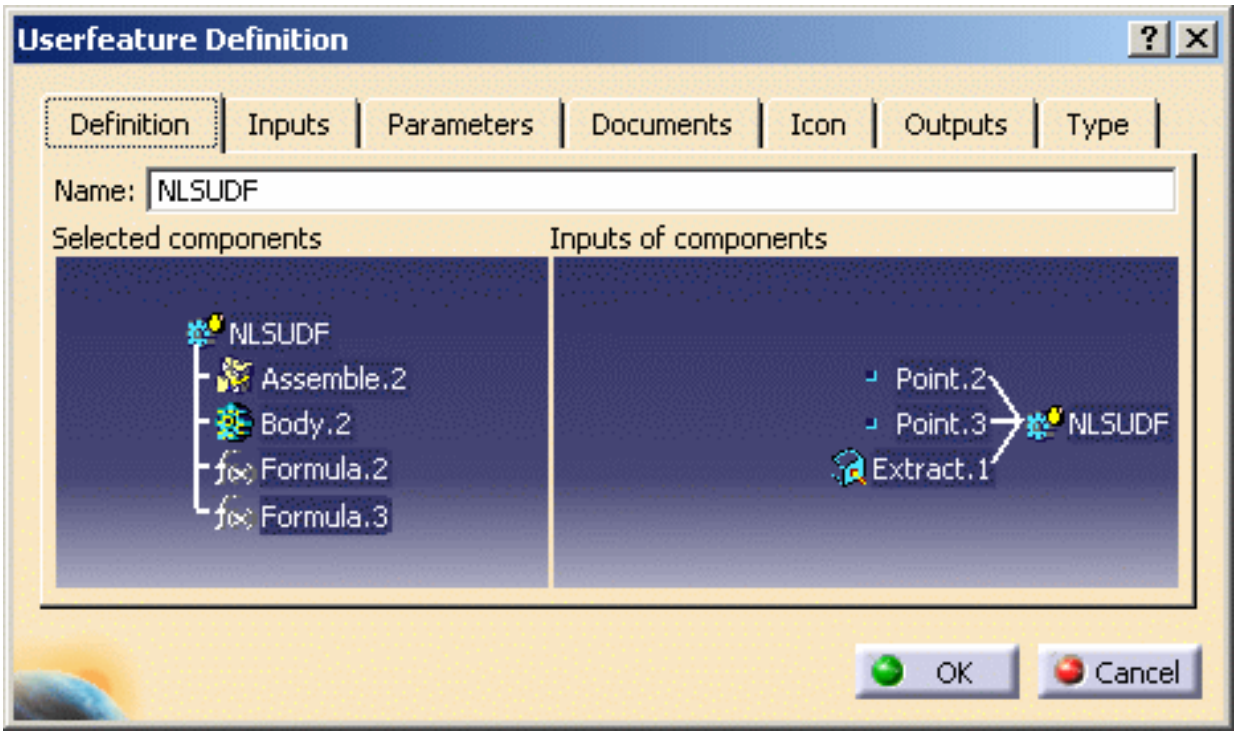
```
Point = "Input point";
Plane = "Support";
Configuration = "Distance configuration";
Configuration.Item1="Short";
Configuration.Item2="Normal";
Configuration.Item3="Long";
Distance = "Wheel distance";
Radius = "Wheel radius";
Fill = "Fill";
//For the Nls message of update error
UpdateErrorMessage = " UPDATE ERROR
MESSAGE IN ENGLISH"
```

The inputs of the .CATGscript file are listed in the opposite cell along with their NLS names: Point = "Input Point".

Note that:

- Inputs names should not contain blank spaces.
- All NLS names are indicated between quotes "" and are separated by ;.
- It is possible to add an error message that will be launched if an update error occurs.
- The name of the file is: CATTypeCATWheel.CATNls

1. From the **Tools->Options** menu, click **Parameters and Measure**, and select the **Language** tab. Click the  button and select the directory that will contain the types file (.CATGScript file). Click **OK** when done.
2. Open the PktcreateaUDF.CATPart file. Note that this file already contains a user feature located below the KnowledgeTemplates node.
3. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench.
4. Click the **Create a UserFeature** icon (). The UserFeature Definition dialog box displays. Replace the default user feature name with NLSUDF, then click the Assemble.2 object in the specification tree. The dialog box now looks like the one below:

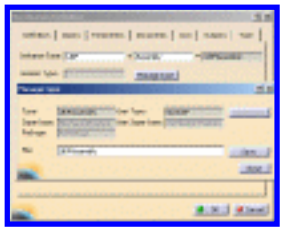


Note that the inputs of the user feature (Point.2, point.3, and Extract.1) will be those used in the .CATNls file.

5. Click the Parameters tab and select the `Body.2\Geometrical Set.1\Circle.2\Circle center radius.1\Radius` parameter. Click **Published Name** and assign it a name: Cylinder_Radius.

6. Assign a type to the user feature. To do so, proceed as follows:

   - Click the **Type** tab.
   - In the Instance Type field, enter UDF and Assembly, and hit the **Enter** key.
   - Click the **Manage type** button.
   - In the Manage Type window, click the **Create type** button. (Click the graphic opposite to enlarge it.) The type is created as well as the associated .CATGScript file which is saved in the directory you previously selected (Step 1.)



   - Click **Save** and **Close** when done.

- Click **OK** to exit the user feature definition window.

7. Save your file and Close *CATIA*.

8. Open a Text Editor and enter the following text to create the .CATNls file:

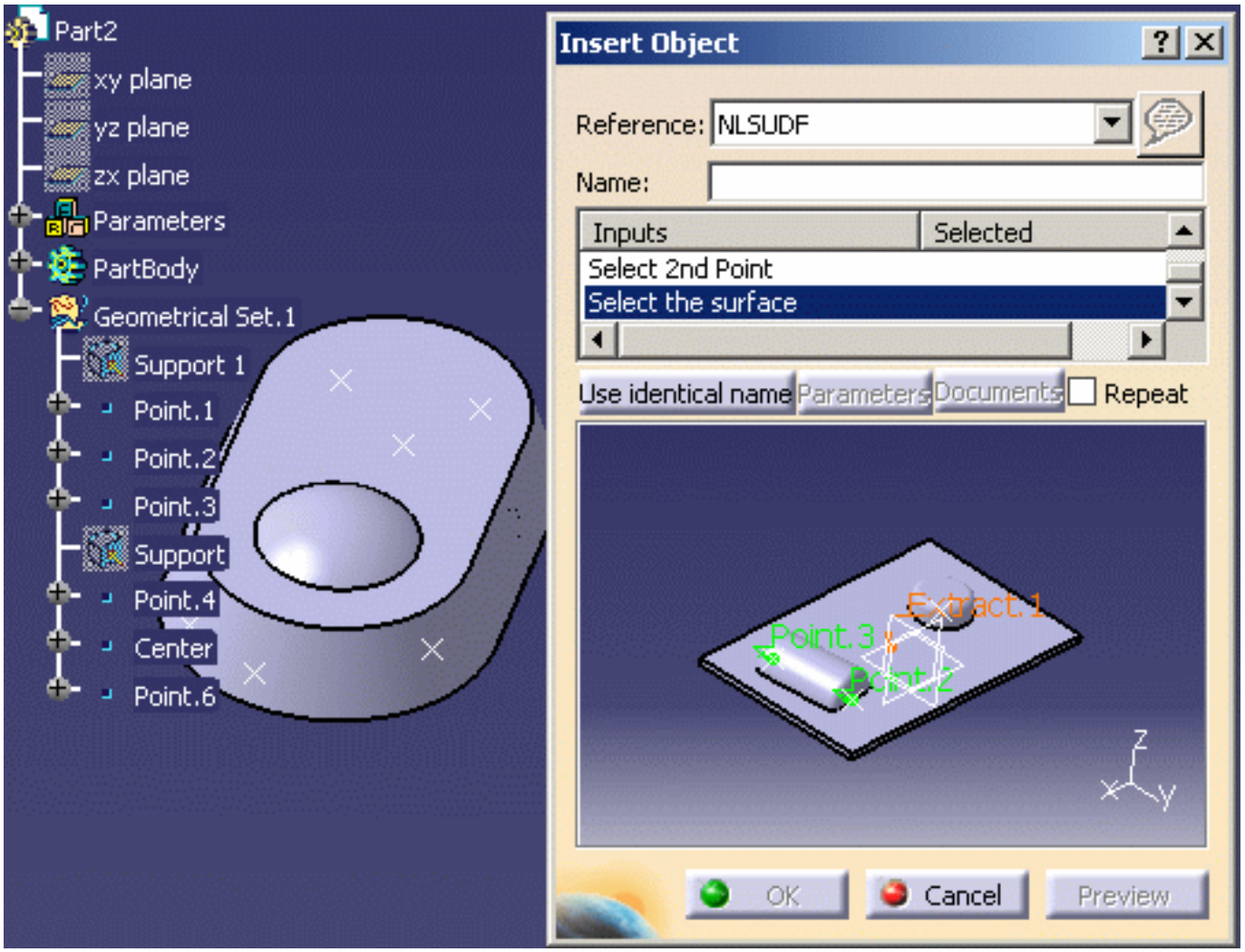
```
Point.2="Select 1st
Point";
Point.3="Select 2nd
Point";
Extract.1="Select the
surface";
Cylinder_Radius="Radius
of the UDF";
```

Note that Point.2, Point.3, and Extract.1 are the inputs of the user feature.

9. Save your file under the following name: CATTypeUDFAssembly.CATNLS in the run-time view. Close the Text Editor.

10. *Open CATIA* and open the PktForInstantiation.CATPart file.

11. In the PKT workbench, click the **Instantiate From Document** icon (). The File Selection window displays. Select the PktcreateaUDF.CATPart file that you have just saved and click **Open**. The following image displays:



12. Select the first point, the second point, and the surface: The user feature is instantiated.

To know more about the user feature definition window, see Introducing the User Feature Definition window. Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on user features.

Saving a User Feature in a Catalog

The task below explains how to store user features in a catalog. This task is not actually a Product Knowledge Template task, but in the context of the Product Knowledge Template product, you will have to carry it out quite often.

You have just created two user features (Pad1 and Pad2). The main interest of user features lies in the instantiation process whereby a user feature stored in a catalog can be reused in a document.

The PktcreatedUDF.CATPart document containing both user features should be open.

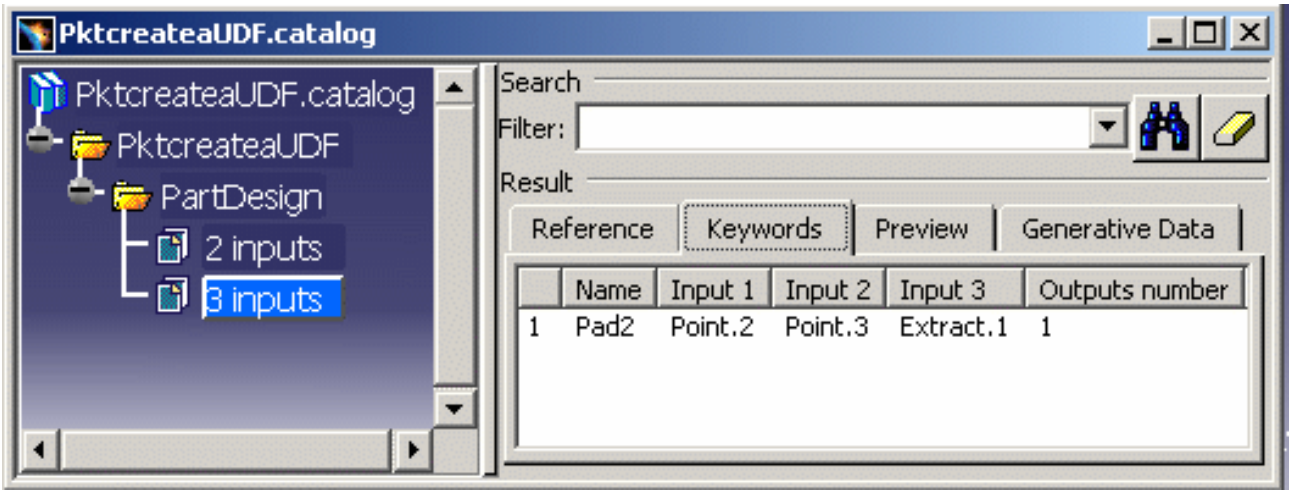
Using the Save Object in a Catalog Command

1. Click the **Save object in a catalog** icon () from the standard menu bar in the PKT workbench. The 'Catalog save' dialog box displays.

   Note that this command is also available from the Part Design and the GSD workbenches.

2. Select the Create a new catalog option and click the button on the right-hand side of the Catalog name field. The dialog box which is displayed allows you to specify a .catalog file where to store the created user features. Enter a file name and click Open. Then click **OK** in the Catalog save dialog box.

3. Open the catalog you have just created (**File->Open** from the standard menu bar). The catalog which is displayed looks like the one below (depending on the name assigned to the catalog):



The left pane displays the two user features created within a tree structure (Pad1 has two inputs while Pad2 has three inputs). Selecting a user feature (3 inputs for example) displays in the right pane the characteristics of the user feature.

About the Reference tab

The user feature name as well as the document it originates from is displayed.

About the Keywords tab

The user feature name as well as its inputs are displayed.

About the Preview tab

The icon you have associated with the user feature (if any) is displayed.

About the Generative Data tab

The resolved queries: A resolved query is relevant for parts with design tables only since it aims at storing a filtered view of the design table data.

To know more about the Catalog Editor, see the *Infrastructure User's Guide*.

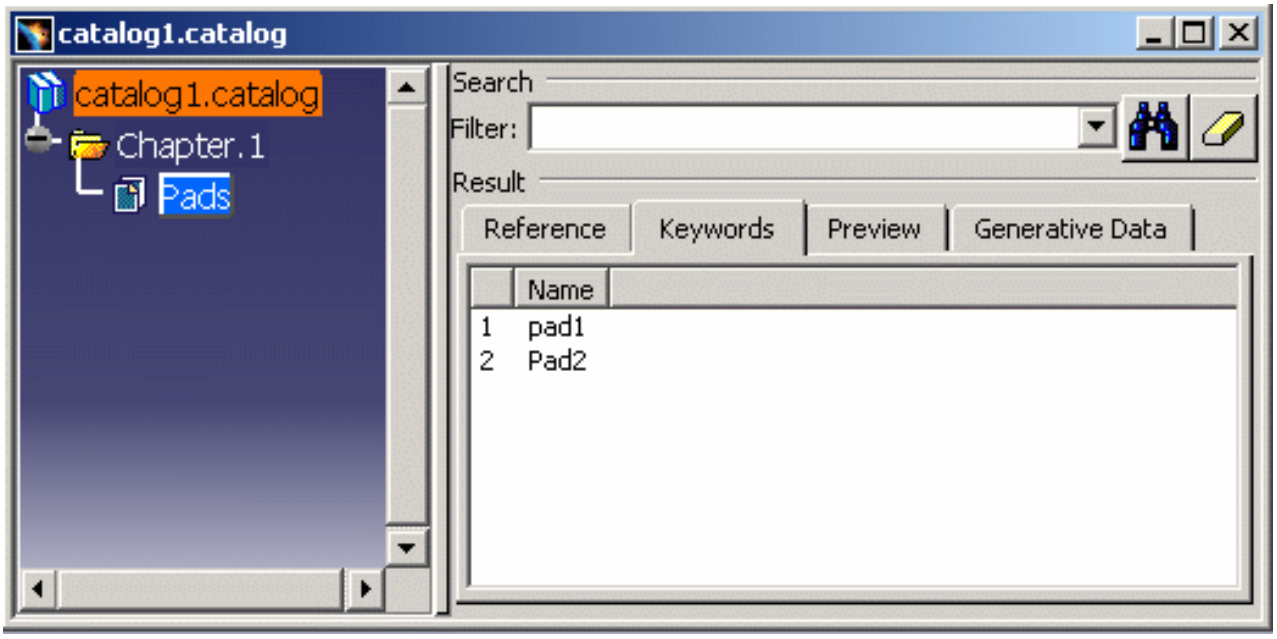
Using the Catalog Editor

1. From the **Start->Infrastructure** menu, access the **Catalog Editor**.
2. Double-click Chapter.1 and click the **Add Family** icon (). The Component Definition Family displays.
3. Change the name of the family: Pads in this scenario and click **OK**.

4. Double-click the Pads family and click the **Add Component** ( ) icon.

5. In the Description Definition dialog box, click the Select external feature button, go back the PktCreateaUDF.CATPart file and select the Pad1 user feature in the specification tree and click **OK**.

6. Repeat the previous step to insert the Pad2 user feature into the catalog.



7. Save your catalog and proceed to the next task: Instantiating a User Feature.

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on user features.

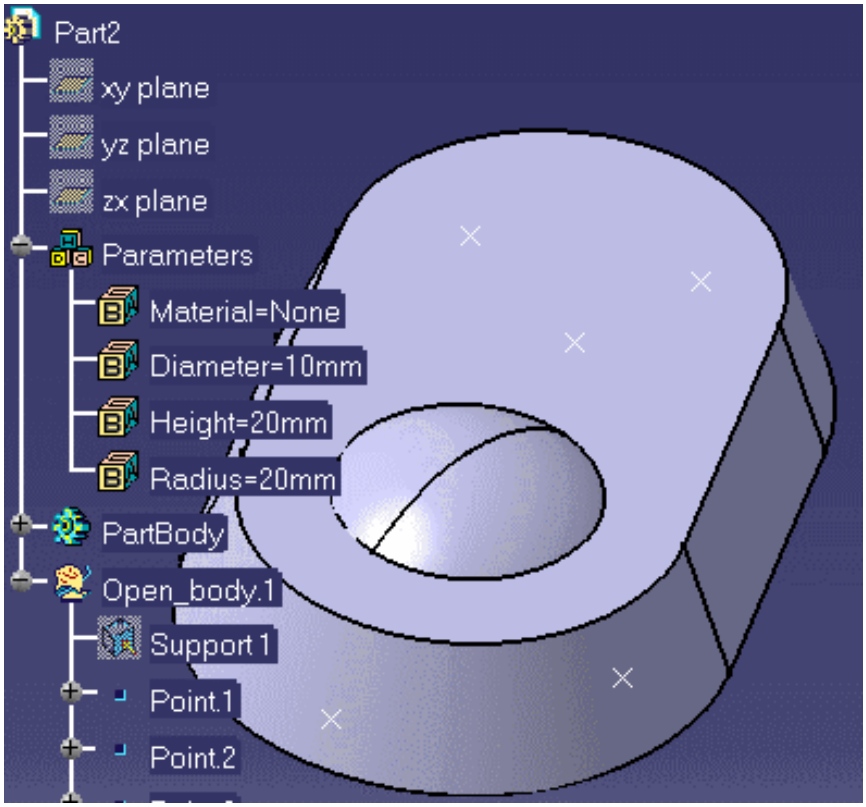
Instantiating a User Feature

The scenario described below shows how to instantiate a user feature

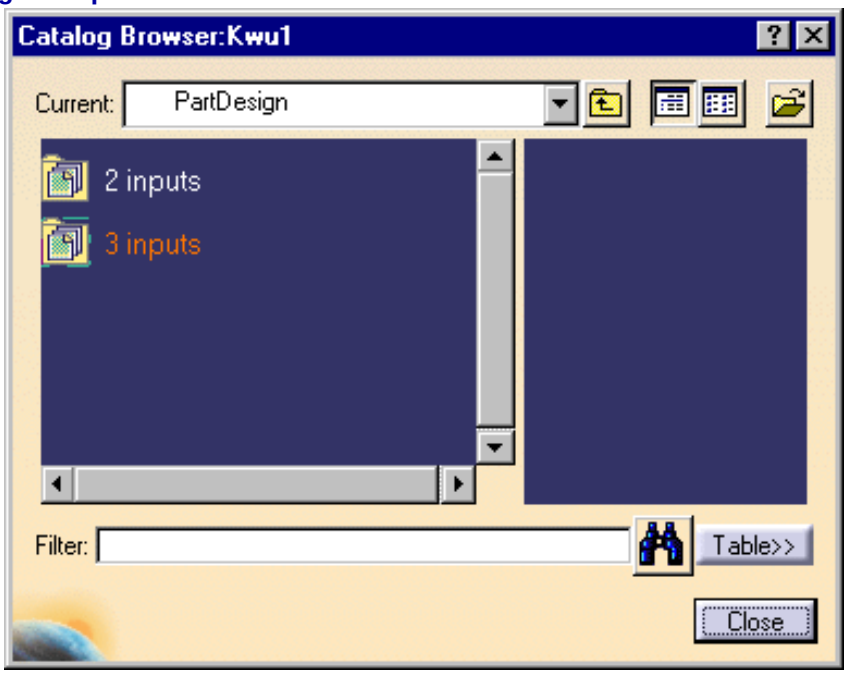
- from a catalog
- from a document containing a user feature and
- from a selection

From a Catalog

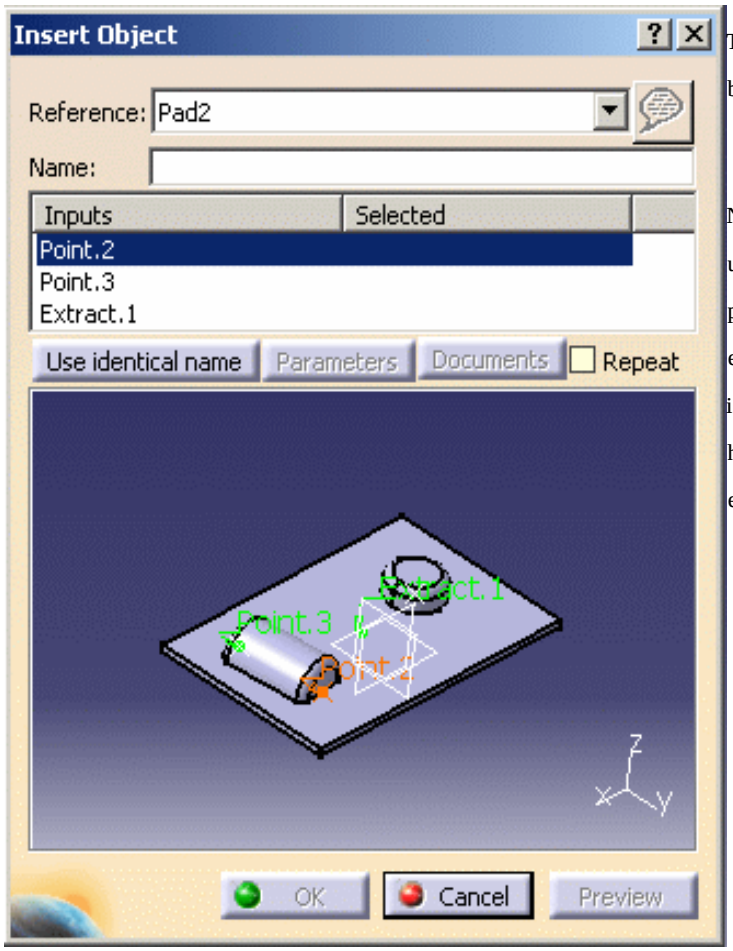
1. Open the PktForInstantiation.CATPart document. The following screen displays.



2. In the standard toolbar, click the  icon. The catalog browser is displayed.
3. Click the  icon. In the dialog box which is displayed, select the catalog containing the user features you want to instantiate. Click Open to open the selected catalog. The dialog box which is displayed next enables you to navigate through the chapters and the families of the catalog until you can access the desired user feature.



4. Double-click the '3 inputs' object and the 'Pad.2' object. The Insert Object dialog box is displayed.



To know more about the Insert Object dialog box, click here.

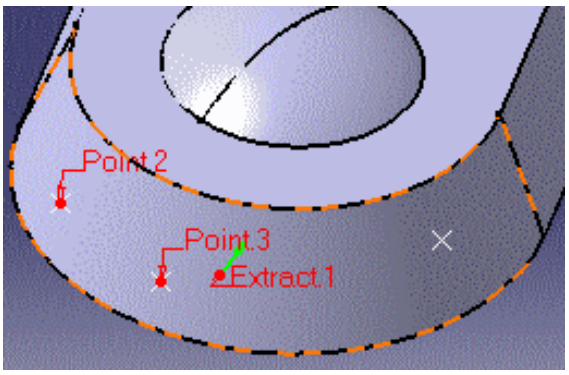
Note that in some cases, when instantiating a user feature, the replacing element does not present the same sub-elements as the replaced element. Therefore you need to clearly indicate in a specific dialog box, the Replace Viewer, how to rebuild the geometry from the replacing element.

5. To instantiate Pad2 into the document, proceed as follows:

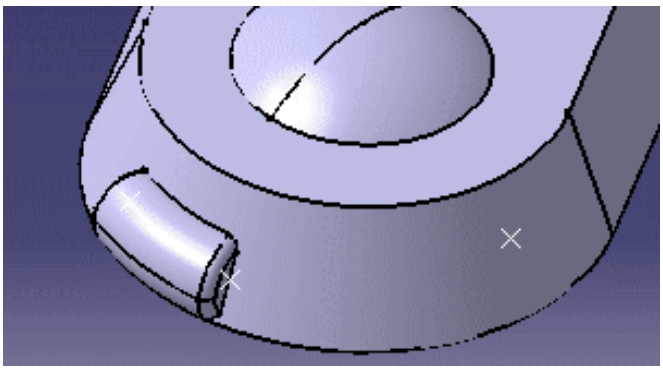
   a. If need be, select Point.2 in the Insert Object dialog box, then select the Point.2 object in the document geometry area or in the specification tree.

   b. Select Point.3 in the Insert Object dialog box, then select the Point.3 object in the document geometry area or in the specification tree.

c. Select Extract.1 in the Insert Object dialog box, then select the face highlighted on the graphic below.



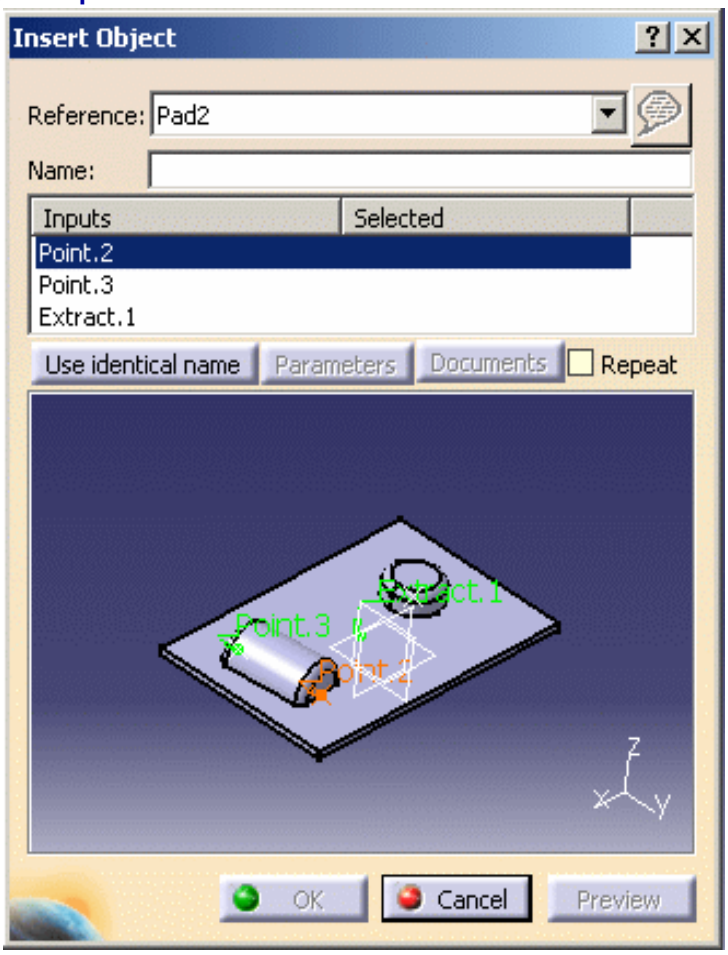
6. Click OK to instantiate the Pad2 user feature and exit the **Insert Object** dialog box. The user feature Pad2 is instantiated into the document. This is what you can see on screen.



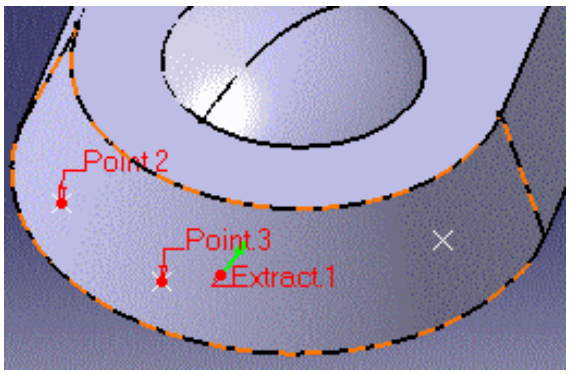
From a Document

1. Open the PktForInstantiation.CATPart document.
2. Click the **Instantiate an element stored in a document** icon ( ). The **File Selection** dialog box displays.
3. Select the PktInstantiateUDFfromDocument.CATPart file and click **Open**.
4. The Insert Object dialog box displays.

- In the Reference scrolling list, select the user feature that you want to instantiate (Pad2 in this scenario).
- If need be, select Point.2 in the Insert Object dialog box, then select the Point.2 object in the document geometry area or in the specification tree.



- Select Point.3 in the Insert Object dialog box, then select the Point.3 object in the document geometry area or in the specification tree.
- Select Extract.1 in the Insert Object dialog box, then select the face highlighted on the figure below.

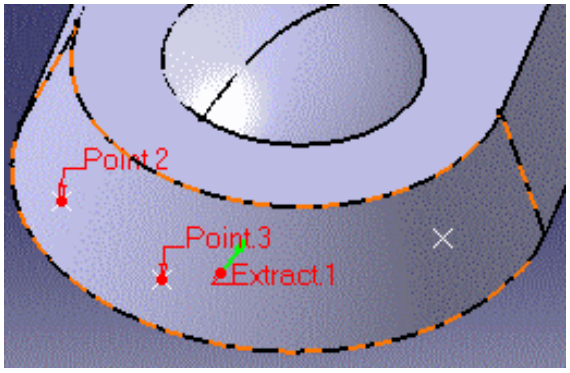


5. Click **OK** when you are done. The user feature is instantiated.

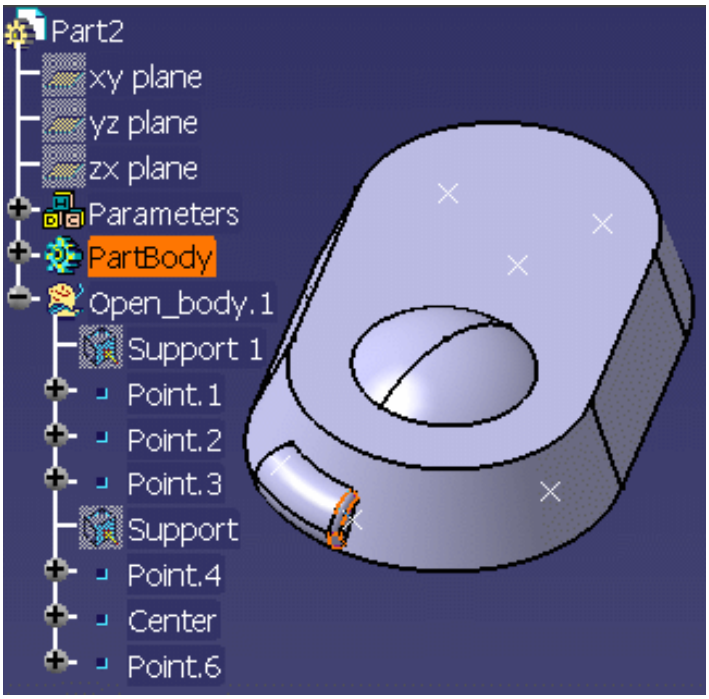
Click here to know more about the Insert Object Dialog box.

From a Selection

1. Open the PktInstantiateUDFfromDocument.CATPart and the PktForInstantiation.CATPart files.
2. Tile the window vertically.
3. Expand the KnowledgeTemplates node in the PktInstantiateUDFfromDocument.CATPart file and click the Pad2 user feature.
4. Go to the PktForInstantiation.CATPart file and click the Instantiate from Selection icon ( ). The Insert Object dialog box displays.
5. Click the **Use identical name** button and click the face highlighted in the picture below.



6. Click **OK** when done. The user feature is instantiated.



Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on user features.

Assigning a Type to a User Feature

This task explains how to reference user features like any other existing types.

To assign a type to a user feature, proceed as follows:

1. Select a Reference Directory For Types in the **Tools->Options->General->Parameters and Measure->Language** tab. This directory will contain the created user types (.CATGScript files). This way, user types will be persistent from a CATIA V5 session to another.
2. Declare your type in the Type tab of the User Defined Feature Definition window.

   We call:

   - User Type, the name attributed to the UDF in the Definition tab.
   - Instance Type, the new type you are declaring.

Note that:

- It is recommended to insert a project prefix in the type name: my_project_hole.
- You can manage your user type by:
  - Storing it in the following CATIA packages: Mechanical Modeler, GSD Package or PartDesign.
  - Assigning it a super type.
- When pushing the button **Create Type**, the new user type becomes available in the session.
- User Features can define new types of objects created by the user and can therefore be searched for like any other type. They are also available in the Knowledge Expert browser.
- If you want other users to use the user feature you created, you will have to provide them with the user feature, the catalog in which it is stored (if stored in a catalog), and the CATGScript file.

1. Open the Pktudfcreateatype.CATPart document.

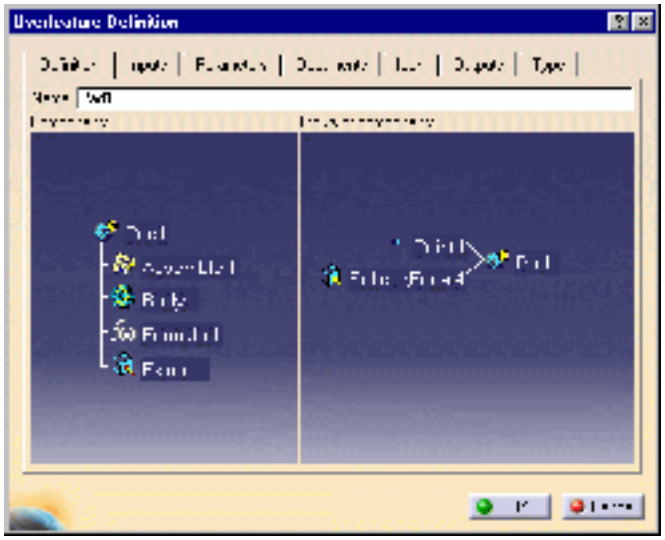
   Pay attention to the Assemble.2 object. This object is the one we are going to use to create a user feature.

2. Select the **Insert->Userfeature->Userfeature Creation...** command from the standard menu bar if you are currently working with the Part Design or Generative Shape Design workbenches or click the **Create a User Feature** icon () if you are in the PKT workbench.

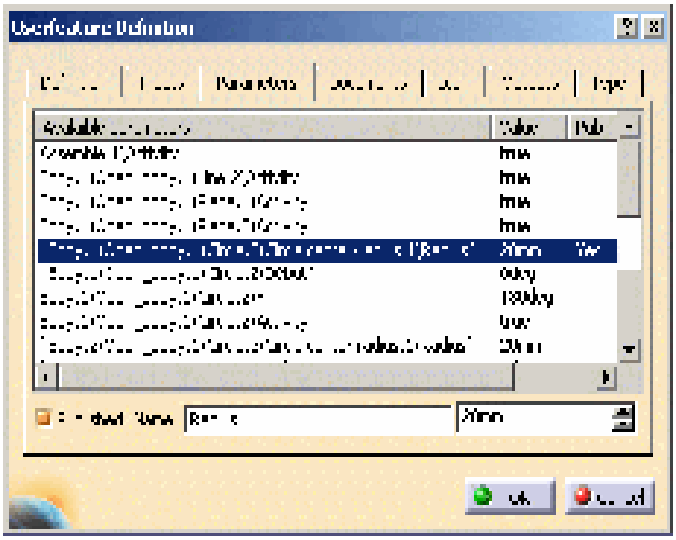
3. The Userfeature definition window opens.

   a) In the **Definition** tab, replace the default user feature name (enter Pad2 as a new name for example) then select the Assemble.2 object in the specification tree.

   (Click the graphic opposite to enlarge it.)



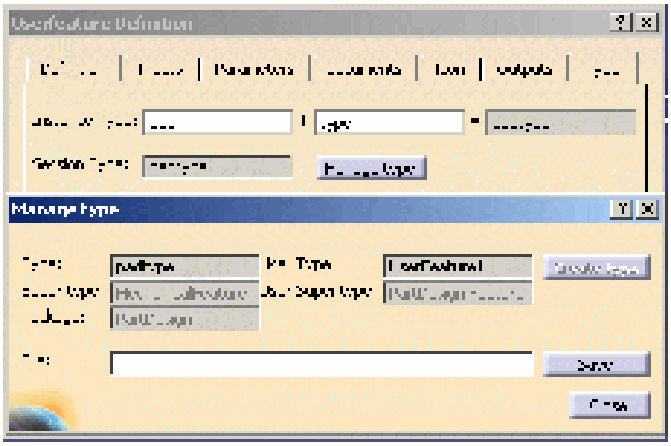
   b) In the **Parameters** tab, publish the parameter that will be published. To do so, proceed as follows:



   (Click the graphic above to enlarge it.)

- Select **`Body.2\Geometrical Set.1\Circle.2\Circle center`** **`radius.1\Radius`**`, click the **Published Name** check box and change the name of the parameter (Radius for example)`

c) In the **Type** tab, enter the name of the instance type:
Enter the first part of the type in the first box, the second part in the second box and hit the **Enter** key.
Click the **Manage Type** button.
Indicate the Super Type and the Package.
Click **Create Type**, **Save**, and **Close**.



(Click the graphic above to enlarge it.)

Note that if you want to publish parameters later, you will have to re-generate the CATGScript in the **Manage Type** window.

- Note that only the MechanicalModeler, the GSD and the Part Design packages are available here.
- Part Design is the default package.

4. Click **OK** to exit the user feature dialog. The Pad2 user feature is added to the specification tree right below the KnowledgeTemplates node. Click here to display the part containing the generated part and here to open the generated .CATGScript file.

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on user features.

Referencing User Features in Search Operations

This task explains how to reference user features like any other existing types and how to perform search operations on these types.

- User Features can define new types of objects created by the user and can therefore be searched for like any other type.
- If you want other users to use the user feature you created, you will have to provide them with the user feature, the catalog in which it is stored (if stored in a catalog), and the CATGScript file.

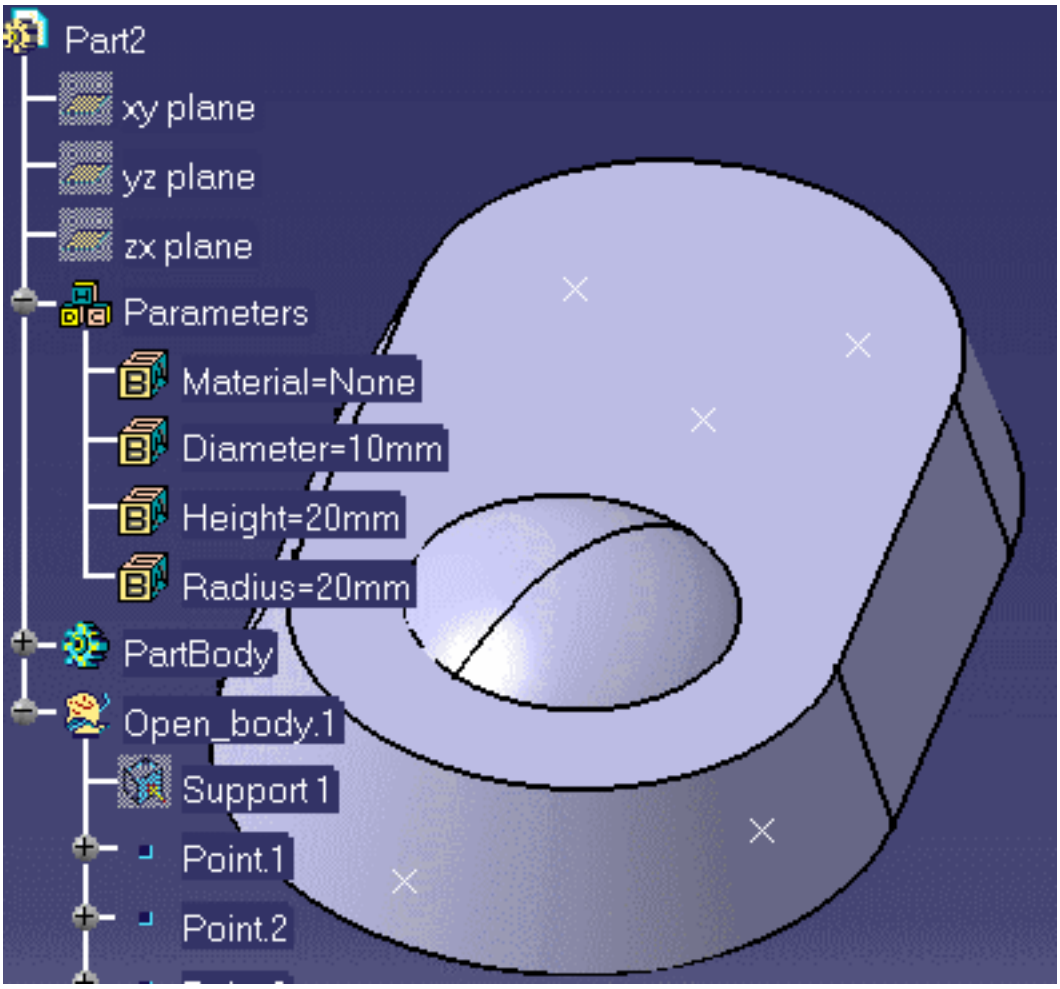
Prior to carrying out this scenario, indicate the reference directory for types (**Tools->Options->General->Parameters and Measure->Language** tab) and copy the Pktpadudf.CATGScript file into this directory.

1. Open the Pktudfcreatedtype.CATPart document and access the Product Knowledge Template workbench (if needed).

2. Select the **Insert->UserFeature->Save In Catalog...** command from the standard menu bar or click the Save object in a catalog icon (). The 'Catalog save' dialog box is displayed.

3. Select the Create a new catalog option and click the button on the right-hand side of the Catalog name field. The dialog box displayed allows you to specify a .catalog file where to store the created user features. Enter a file name and click Open. Click OK in the Catalog save dialog box.

4. Open the PktForInstantiation.CATPart document. The following screen is displayed.



5. In the standard toolbar, click the  icon. The catalog browser is displayed.

6. Click the  icon. In the dialog box which is displayed, select the catalog which contains the user features you want to instantiate. Click Open to open the selected catalog. The dialog box displayed next depends on your last interaction on this catalog.

   Double-click the object displayed in the left pane until you get Pad2 on screen:

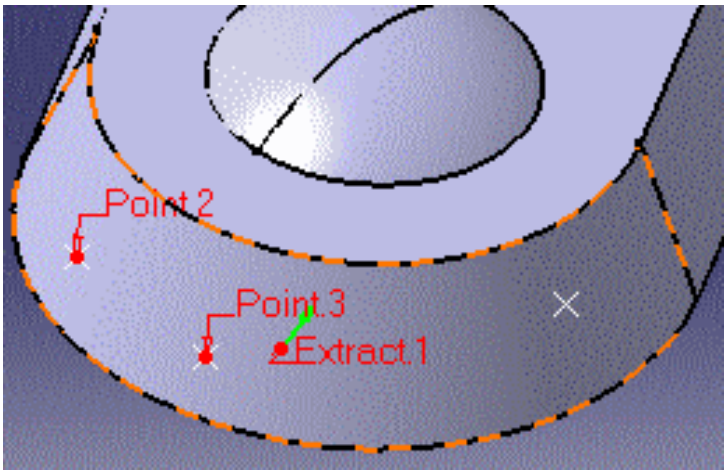
12. Double-click the '**Pad2**' object. The **Insert Object** dialog box is displayed.

(Click the graphic opposite to enlarge it.)

To know more about the Insert Object dialog box, click here.

7. Instantiate Pad2 in the document. To do so, proceed as follows:

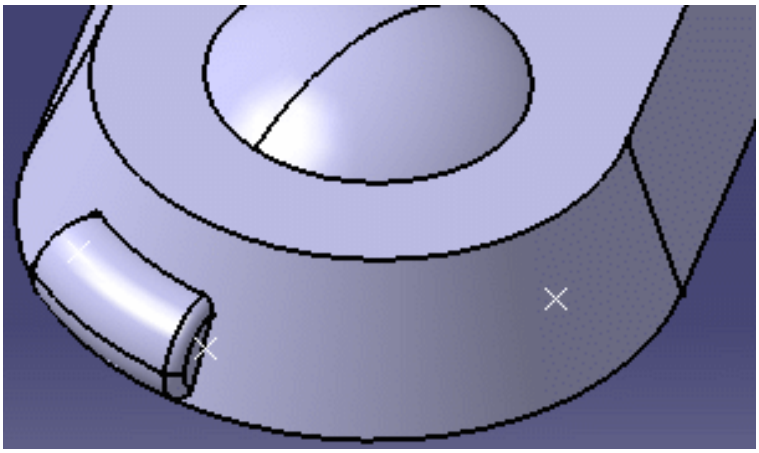
a. Select Point.2 in the "Insert Object" dialog box, then select the Point.2 object in the document geometrical area or in the specification tree.



b. Select Point.3 in the "Insert Object" dialog box, then select the Point.3 object in the document geometrical area or in the specification tree.

c. Select Extract.1 in the "Insert Object" dialog box, then select the face highlighted on the figure below.

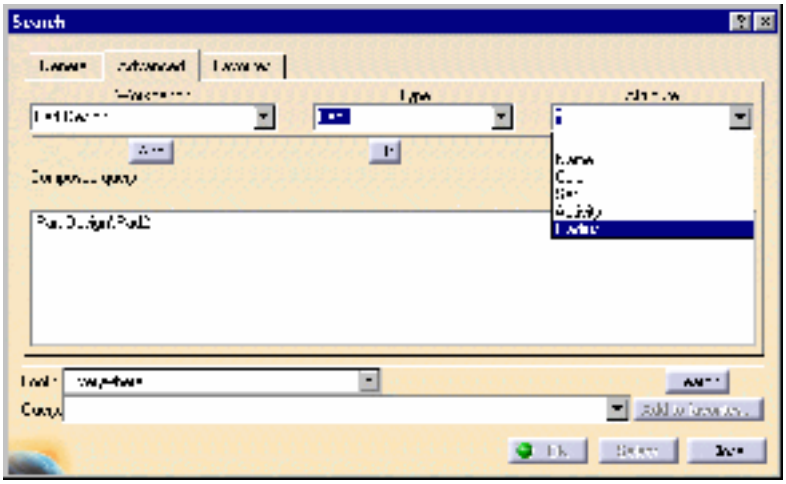
d. Click **OK** and Close. Pad2 is instantiated.



8. Select the **Edit->Search** (**CTRL+F**) command. The **Search** window opens.

Select the Advanced tab.

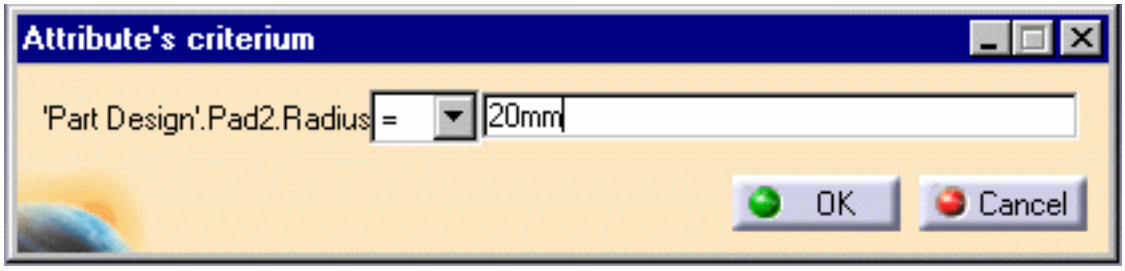
(Click the graphic opposite to enlarge it.)



9. Select **Part Design** under **Workbench**.

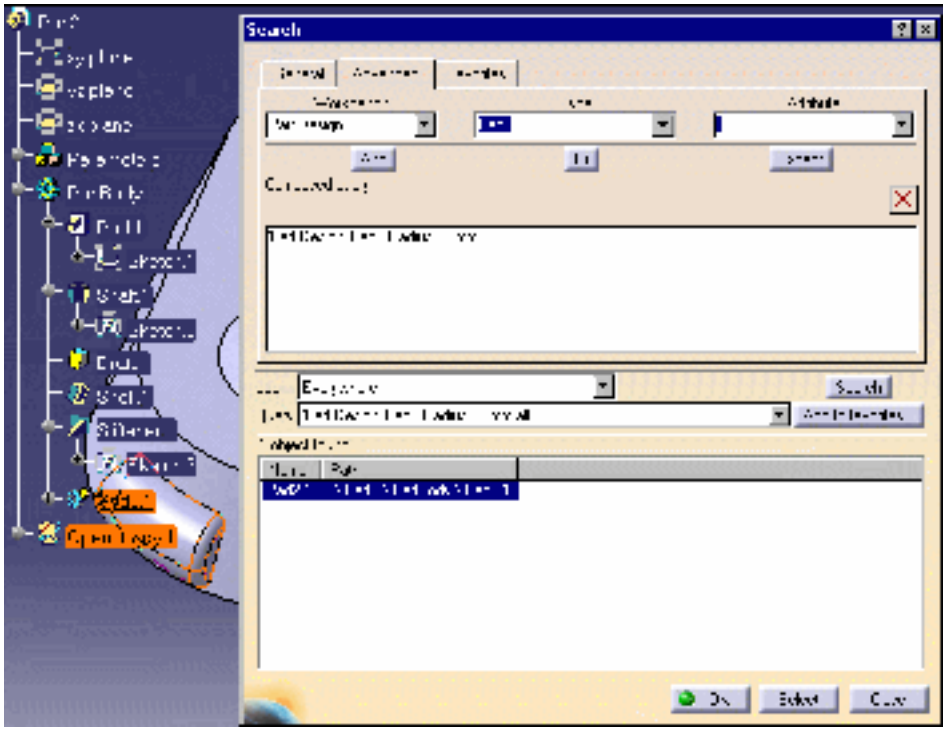
10. Select **Pad2** under **Type**. Pad2 is the Definition name defined in the Definition tab.

11. Select **Radius** under **Attribute**. The **Attributes' criterium** dialog box opens. Enter 20mm in the = field. Click **OK**.



Note that Pad1 and Pad2 are now considered like any other types and can therefore be searched for.

12. Click **Search**: the Pad2 instance (Pad2.1) is displayed in the Object found field and is highlighted both in the specification tree and in the geometrical area (click the graphic below to enlarge it.)



Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on user features.

User Features: Useful Tips

Creating a User Feature

Note that the limitations that apply to PowerCopies also apply to user features.

- As far as possible, minimize the number of elements making up the user feature.
- When defining user features including sketches, use profiles constrained with respect to edges or faces rather than to planes. Additionally, set the option Create geometrical constraints off before sketching. Generally speaking, it is always preferable to use profiles both rigid and mobile.
- It is preferable to constrain elements with respect to external references such as faces, edges, reference or explicit planes.
- It is preferable not to use projections nor intersections in your sketch if you want to use your sketch in a user feature.
- Avoid constraints defined with respect to reference planes.
- Before creating your user features, make sure that your sketch is not over-constrained.
- Make sure that your sketch is iso-constrained (green color). You can use non-iso-constrained sketches, but it will be more difficult to understand and control the result after instantiation.
- To create a User Feature, create first a PowerCopy, and try it in different contexts. When the instantiation is OK, create the User Feature by selecting the PowerCopy. It is easier to understand and modify a PowerCopy.
- Provide basic and full User Feature on the same geometry (with or without final Trim for example). If an update error occurs, the user can try the basic User Feature and perform manually the last operations.

Managing inputs:

- Always rename your inputs to help the end user understand what he needs to select.
- A formula is automatically included in a user feature definition when all its parameters are included. Otherwise, i.e. if at least one parameter is not selected as part of the user feature, you have to manually select the formula to make it part of the definition. If you do so, all the formula's parameters that have not been explicitly selected, are considered as inputs of the user feature.
- Note that when including parameters sets containing hidden parameters in a user feature, the hidden parameters are automatically instantiated when instantiating the user feature.

Preview:

- In a Part document, create only one User Feature reference. It is not a technical restriction, but there are at least two raisons for this: The cost of an instantiation will be smaller if the Part document is smaller. The end user can more easily understand the feature to be instantiated.
- Put in "show" only the input and the result (to help the end user to understand what he needs to select).
- Use color to differentiate inputs (put transparency on result for example).
- Choose a pertinent viewpoint before saving the Part document reference, default viewpoint in preview during instantiation will be the same.

Geometry:

- Create sketches on an axis system, in order to better control the Sketch position.
- Avoid constraining your 2D elements with respect to HV absolute axis. The result you obtain after instantiating the PowerCopy could be unstable. Actually, you cannot control the position of the origin of the absolute axis nor its orientation.

Catalog:

- Do not forget catalog integration if you want to provide several User Features.

Instantiating a User Feature

- Always check the orientation for curve and surface.
- If you need to instantiate a user feature several times on the same input, rename your inputs and use the "Use identical name" option.

User Features: Limitations

- Note that datums (features that cannot be calculated) cannot be inputs of user features.
- Note that sub-elements cannot be inputs of user features. For example, the face of a pad cannot be an input.
- Note that when creating the user feature, it is not possible to edit (add/remove) inputs after leaving the **Definition** tab. Click the **Cancel** button and create a new user feature.
- UDF graphical properties (such as color, show/hide status, ...) depend on the graphical properties of its components at creation. As soon as the UDF is created, i.e. as soon as the components are defined and validated (either by clicking **OK** in the definition panel or by changing tabs), the graphical properties of the UDF are "frozen" and thus independent from the graphical properties of its components.
  The reason why the UDF graphical properties are independent from its internal graphical properties is that the UDF is a feature with its own graphical properties. Those properties can be modified using the Properties contextual command. If the UDF properties were dependant from the UDF internal components, the user wouldn't be able to modify the UDF graphical properties using the Properties contextual command, or the graphical properties available from the contextual command and graphical properties defined by parameter would not match.
  So it is highly recommended not to use Knowledge parameters inside the UDF to drive its graphical properties.

To know more about User Features ...

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on user features.

The **UserFeature** command can be accessed by selecting the **Insert->UserFeature** command from the following workbenches:

- Part Design
- Generative Shape Design

and by clicking the **Create a User Feature** icon () from the Product Knowledge Template workbench.

A User Feature is a template that works at the part level. From a collection of features (geometry, literals, formulas, constraints, etc.), the user can create his/her own feature. The result is a Part Design feature or a Shape Design feature that can be reused in the design of another part. The created feature can be saved in a catalog.

A user feature:

- Allows to create applicative features
- Allows to hide design specifications and preserve confidentiality (for instance to sub-contractors)

User features (like a line for Drafting or a check for Knowledge Advisor) are open and shareable objects. This capability significantly increases the potential application of user features since it enables to:

- Find user features by attributes.
- Generate user features with the Scripting language to simplify the process of creating scripts .
- Define expert rules working on user features with Knowledge Expert (to know more, see the *Knowledge Expert User's Guide*).
- Use user features in Knowledge Advisor reactions.
- Develop CAA functions based on user defined variables.

Managing Part and Assembly Templates

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on document templates. Refer to To know more about Part and Assembly Templates to know more about these features.

Create a Document Template: Select the **Insert -> Document Template Creation ...** command or click the **Create Document Template** icon ( ) (if in the PKT workbench), select the elements making up the document template from the specification tree, define a name for the document template and its reference elements then choose an icon for identifying it.

Introducing the Document Template Definition Window
Creating a Part Template
Instantiating a Part Template
Adding an External Document to a Document Template
Document Templates: Methodology
To Know More About Part And Assembly Templates

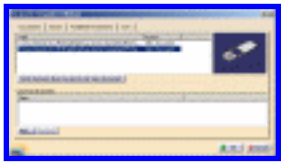
Introducing the Document Template Definition Window

The **Document Template Definition** window can be accessed by selecting the **Insert->Document Template Creation...** command from the following workbenches:

- Product Structure
- Part Design
- Assembly Design
- Generative Shape Design
- Wireframe and Surface Design

The user can access the Product Knowledge Template workbench from the Part Design and the Product Structure workbenches.

The Documents tab



The **Documents** tab shows the complete path and Action of the files referenced in the Template. The Action status can be either:

- Same Document or
- New Document.

If the document is seen as New Document, it is then duplicated and does not have any link with the original component (equivalent of the **New from...** command.)

If the document is seen as Same Document, a link is maintained with the original file.

The 

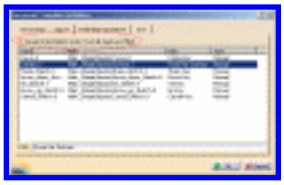
button enables you to modify the Action of the components.

The 

buttons of the External documents sections enable you to select external documents and insert them into the template.

It is now possible to associate non CATIA (ENOVIA VPM V5, ...) documents to a template. 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.

The Inputs tab



The **Inputs** tab enables you to define the reference elements making up the Template by selecting them in the geometry or in the specification tree.

The **Accept instantiation even if not all inputs are filled** option enables users to determine if the template can be instantiated even if not all inputs are valuated. If all inputs are not valuated, old inputs will be kept and isolated at instantiation. This option can be useful if there is more than one way to position the template in context, if you want all these combinations to be available but you want to use only one of them at the same time. To see an example, see Creating a Part Template and lnstantiating a Part Template.

For a clearer definition, you can select these items in the viewer and enter a new name in the **Role** field.

The **Role** field enables you to select one of the items displayed in the window and to rename it. It is used at instantiation through the Use identical name button in the Insert object panel.

The **Type** column indicates if the input is manual or automatic. The inputs are considered as

- **Manual** if they are added manually
- **Automatic** if they are external references that point an object defined outside the template.

The Published Parameters tab

The **Published Parameters** tab enables you to define which parameter value used in the Template you will be able to modify when instantiating it.

The **Edit List...** button enables you to access the list of parameters, and to select those you want to publish. These parameters are displayed in the Part Numbers viewer.

The **Auto modify part numbers with suffix** check box, if checked, automatically modifies the part numbers at instantiation if the part numbers already exist.

- Note that if the user wants to manage the way part numbers are modified at instantiation, he just needs to uncheck this option and click, at instantiation, the Parameters button in the Insert Object dialog box. This way he can access the part numbers that he wants to modify.

- The unicity of part numbers is now ensured when instantiating document templates into different documents or when the document template is used by different users. When the part numbers renaming mode is set to automatic, a suffix parameter is automatically published by the document template. At instantiation, after valuating the inputs of the document template, suffixes can be changed by clicking the Parameters button in the Insert Object window. Note that it is not possible to "unpublish" the suffix or to change its role.

The Icon tab

The **Icon** tab enables you to modify the icon identifying the Template in the specifications tree. A subset of icons is available when clicking the **Icon choice** button.

Clicking **...** displays the Icon Browser, showing all icons loaded in your *CATIA* session.

The **Grab screen** button enables you to capture an image of the template to be stored along with its definition.

The **Remove preview** button enables you to remove the image if you do not need it.

The assembly structure of the documentation template should not be modified after the document template definition (you cannot add or remove documents for example.)

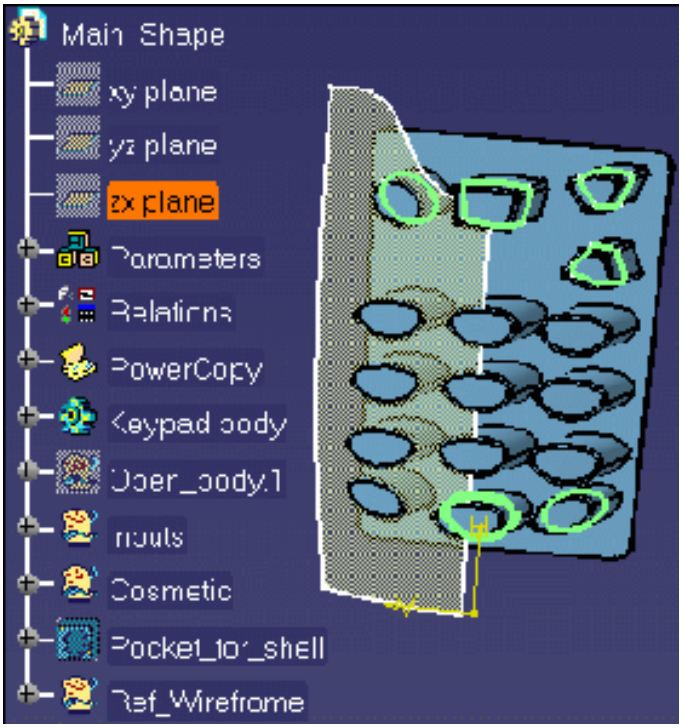
Creating a Part Template

This scenario explains how to create a part template containing a keypad that will be instantiated into a CATProduct document. In this scenario, the user:

- Creates 2 document templates. When creating the first document template, he does not check the **Accept instantiation even if not all inputs are filled** option (Steps 1 to 4). When creating the second document template, he checks the **Accept instantiation even if not all inputs are filled** option (Steps 5 to 8). To know more about this option, see Introducing the Document Template Definition Window.
- Saves both document templates in a catalog.

Creating the first template

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

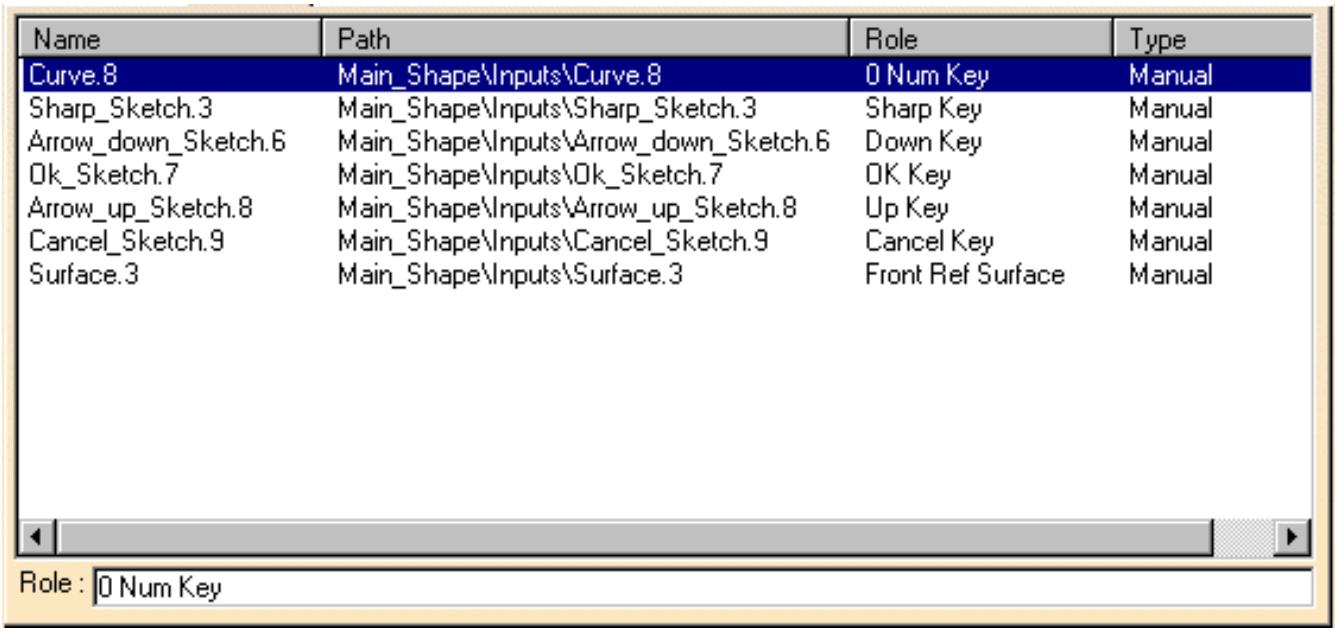


2. From the **Insert** menu, select the **Knowledge Templates->Document Template ...** command (in the Part Design workbench) or, if in the Product Knowledge Template workbench, click the **Create a Document Template** icon ( ). The **Document Template Definition** window displays.
3. In the **Document Template Definition** window, click the **Inputs** tab to select the inputs. To do so, proceed as follows:

- In the geometry, select the following features:

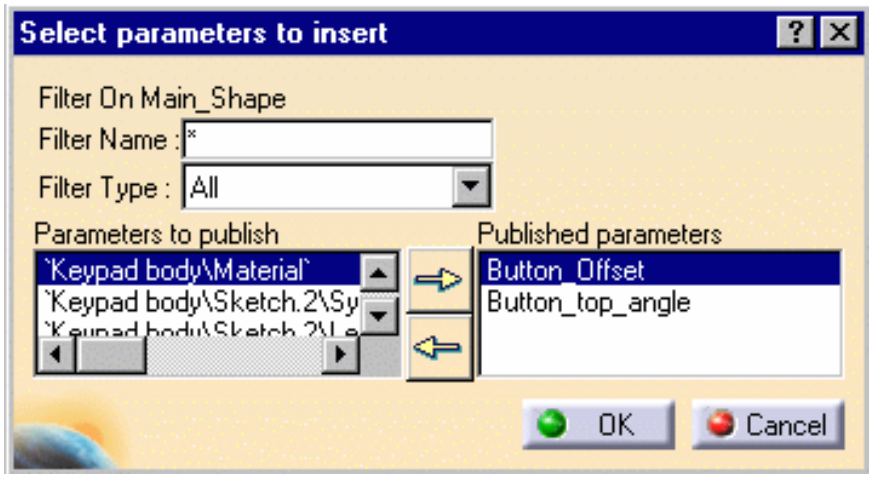
- Curve.8
- Sharp_Sketch.3
- Arrow_down_Sketch.6
- Ok_Sketch.7
- Arrow_up_Sketch.8
- Cancel_Sketch.9
- Surface.3

- In the **Inputs** tab, select the Curve.8 feature and assign it a role in the **Role** field. Repeat the same operation for the features you selected. The final Inputs tab should look like the picture below.



4. Click the **Published Parameters** tab to publish parameters. To do so, proceed as follows:

- Click the Edit List... button. The **Select parameters to insert** window displays.



- Use the arrow to select the **Button_Offset** and the **Button_top_angle** parameters in the **Parameters to publish** column.

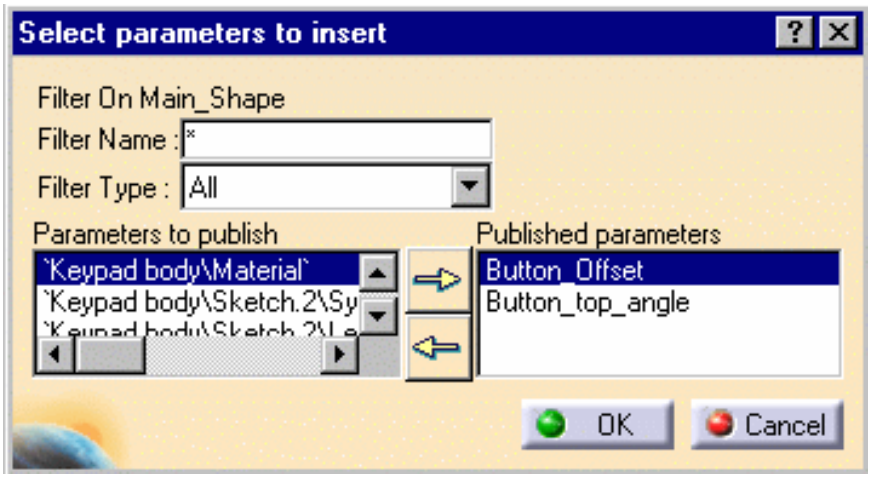
- Click **OK** twice. The Document template is added to the KnowledgeTemplates node.

- Right-click DocumentTemplate.1 and select the **Properties** command to rename the document template.
- In the Feature Name field, enter Keypad1. Click **OK** to validate.

Creating the second template

1. From the **Insert** menu, select the **Knowledge Templates->Document Template ...** command (in the Part Design workbench) or, if in the Product Knowledge Template workbench, click the **Create a Document Template** icon ( ). The **Document Template Definition** window displays.
2. In the **Document Template Definition** window, click the **Inputs** tab and select the following inputs in the specification tree:
   - Curve.8
   - Sharp_Sketch.3
   - Arrow_down_Sketch.6
   - Ok_Sketch.7
   - Arrow_up_Sketch.8
   - Cancel_Sketch.9
   - Surface.3
3. Check the **Accept instantiation even if not all inputs are filled** check box.
4. Click the **Published Parameters** tab to publish parameters. To do so, proceed as follows:

- Click the Edit List... button. The **Select parameters to insert** window displays. In the **Parameters to publish** column, click the **Button_Offset** and the **Button_top_angle** parameters and use the arrow to select them.



- Click **OK** twice. The Document template is added to the KnowledgeTemplates node.

- Right-click DocumentTemplate.2 and select the **Properties** command to rename the document template.

- In the Feature Name field, enter Keypad2. Click **OK** to validate.

- Save your file.

5. Store the document template in a catalog. To do so, proceed as follows:

   - If not already in the **Product Knowledge Template** workbench, from the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench.

   - Click the **Save in catalog** icon (). The Catalog save dialog box displays.

   - Click **OK** to create a new catalog or the **...** button to change the name of the catalog. The catalog is created.

   - Click here to display the result catalog file. Click here to display the result .CATPart file.

6. Close your file and proceed to the next task: Instantiating a Part Template.

Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried on Document Templates.

Instantiating a Part Template

This scenario explains how to instantiate a template into a CATProduct file. It is divided into 2 different parts:

- The user instantiates Keypad1, a document template saved in the PktKeypadscatalog.catalog.
- The user instantiates Keypad2, a document template saved in the PktKeypadscatalog.catalog.

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

- PktMobilePhoneSupport.CATProduct that is made up of the following CATPart and CATProduct files:

PktBottomcase.CATPart	PktBattery.CATPart
PktBody.CATPart	PktLens.CATPart
PktIndus.CATPart	PktLCD30-28.CATPart
PktFrontShell.CATPart	PktElectronic.CATProduct
PktPlanarCard.CATProduct	PktSpeaker.CATPart
InteractiveBoard.CATPart	PktCapacitor_500.CATPart
PktCapacitor_700.CATPart	PktChip_AC30.CATPart
PktChip_AC110.CATPart	PktChip_AC20.CATPart

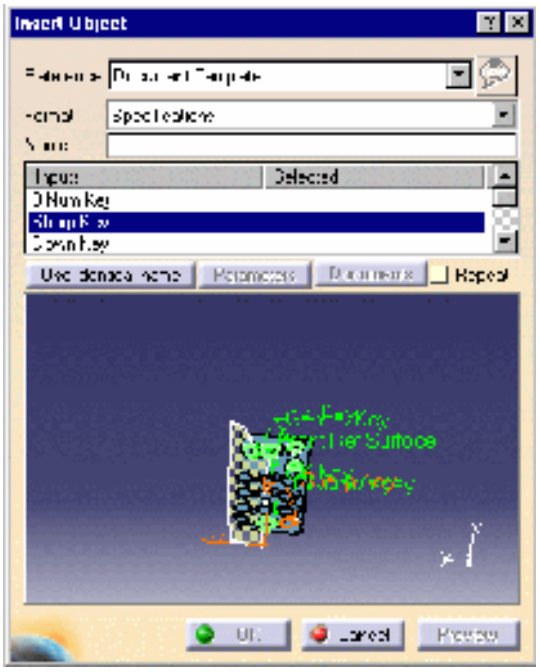
- PktKeypadscatalog.catalog: This catalog contains 2 document templates: Keypad1 and Keypad2. When creating Keypad1, the **Accept instantiation even if not all inputs are filled** option was unchecked. When creating Keypad2, the **Accept instantiation even if not all inputs are filled** option was checked.

Instantiating Keypad1

1. Open the PktMobilePhoneSupport.CATProduct file.

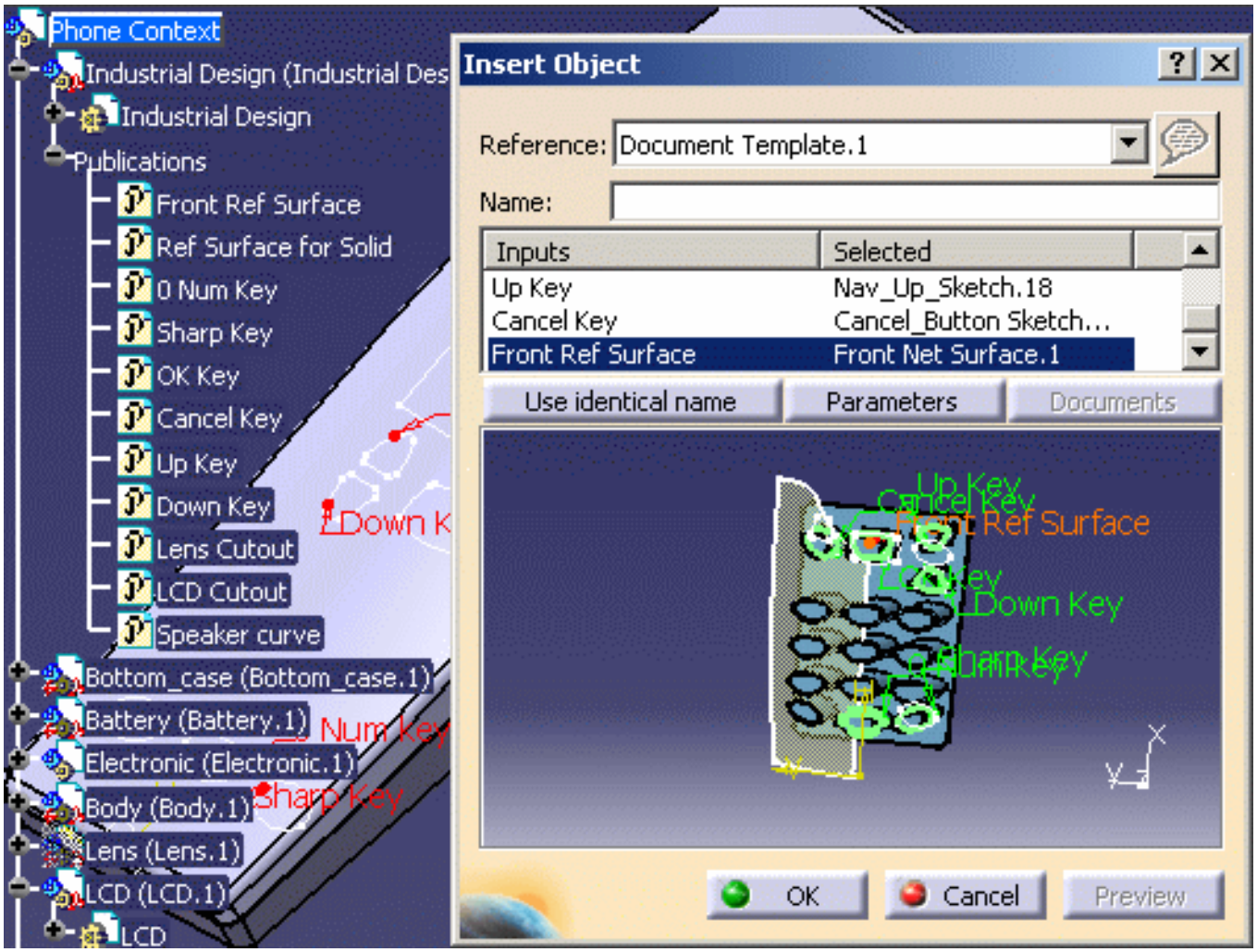
2. Click the **Open Catalog** icon ()and select the PktKeypadscatalog.catalog that you created in the Creating a Part Template topic. The Catalog Browser opens.

3. Double-click DocumentTemplate, 7 inputs and Keypad1. The **Insert Object** window opens. (Click the graphic opposite to enlarge it).

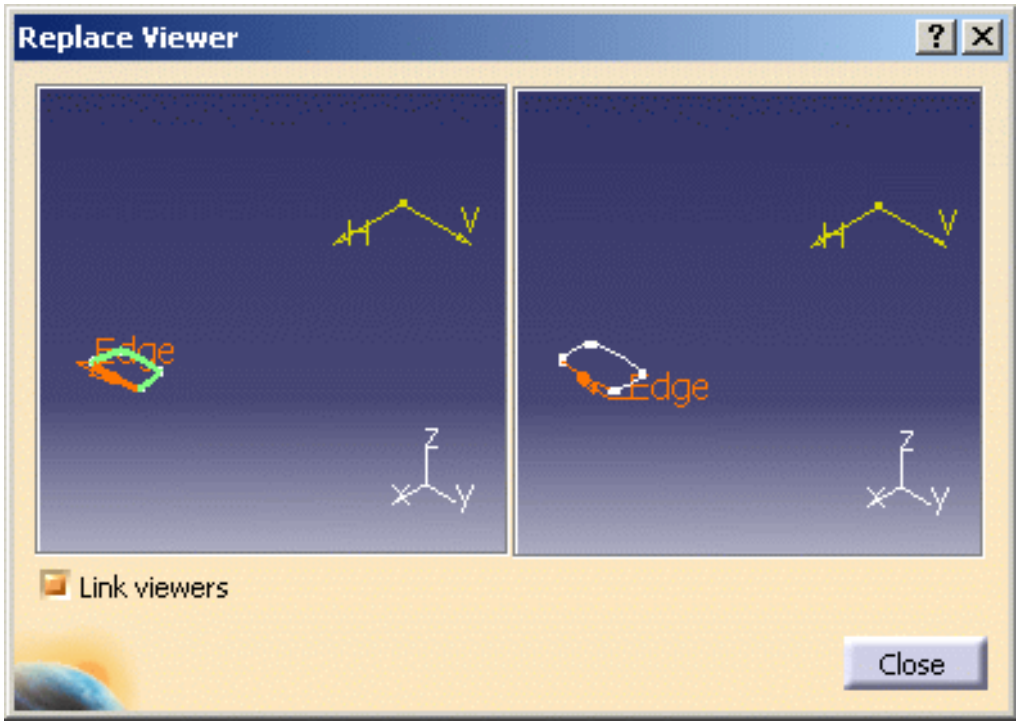


To know more about the Insert Object dialog box, click here.

4. Value the **Inputs** by selecting the publications located below the Industrial Design node in the specification tree or click the **Use Identical Name** button in the **Insert Object** window.



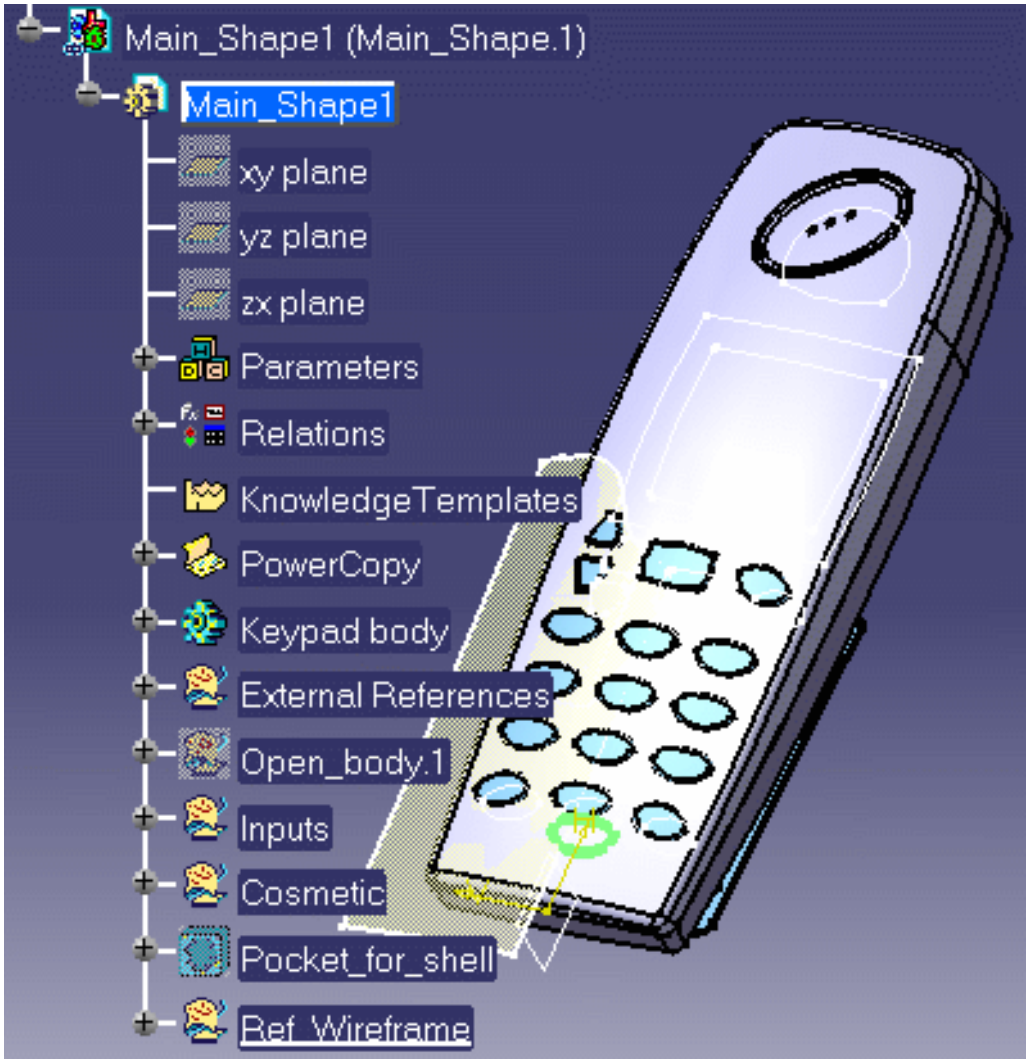
5. Make the appropriate selections in the **Replace Viewer window** (see picture below) and click **OK** when done.



Note that in some cases, when instantiating a part or assembly template, the replacing element does not present the same sub-elements as the replaced element. Therefore you need to clearly indicate in a specific dialog box, the Replace Viewer, how to rebuild the geometry from the replacing element.

6. Click **OK** in the Check warning box, then **Close**. The keypad is instantiated (see picture below.)

7. Close your file.



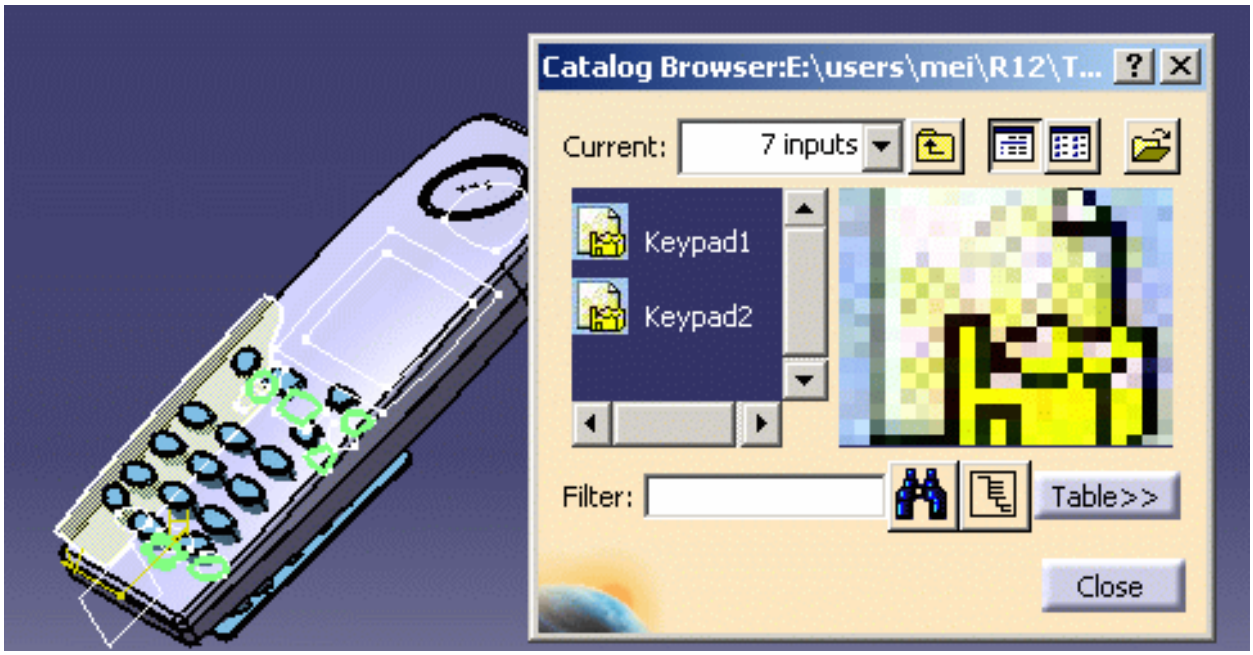
Instantiating Keypad2

1. Open the PktMobilePhoneSupport.CATProduct file.

2. Click the **Open Catalog** icon and select the PktKeypadscatalog.catalog that you created in the Creating a Part Template topic. The Catalog Browser opens.

3. Double-click Document Template, 7 inputs and Keypad2. The **Insert Object** window opens.

4. Click **OK** in the Insert Object window. The keypad is instantiated. Note that you do not have to value the inputs since the **Accept instantiation even if not all inputs are filled** option was checked when creating the Keypad2 part template.



Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried on Part Templates.

Adding an External Document to a Document Template

This task shows how to insert a drawing into a part template and how it is updated at instantiation. The scenario is divided into the following steps:

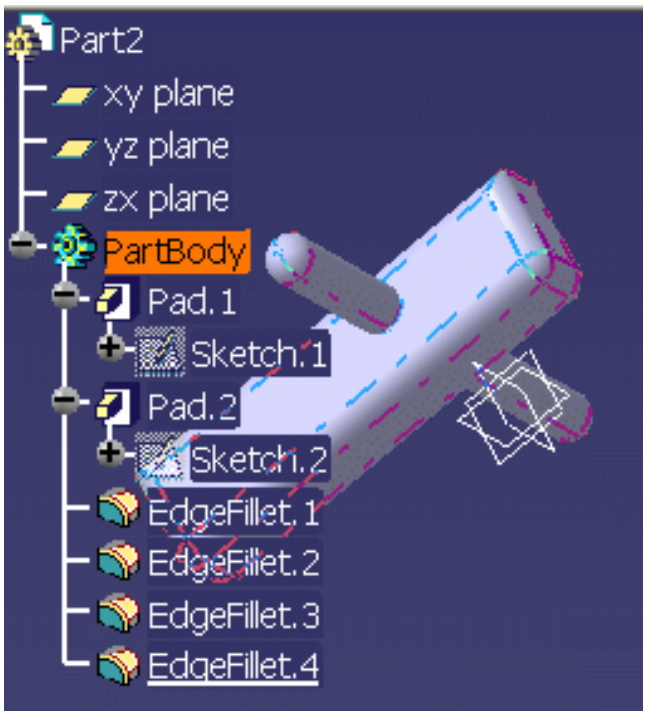
- Creating a drawing from an existing part
- Creating the part template
- Instantiating the part template and updates the generated drawing.

Note that the document(s) that can be added to part and assembly templates must belong to one of the following types:

- .CATDrawing
- .CATProcess
- .CATAnalysis

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

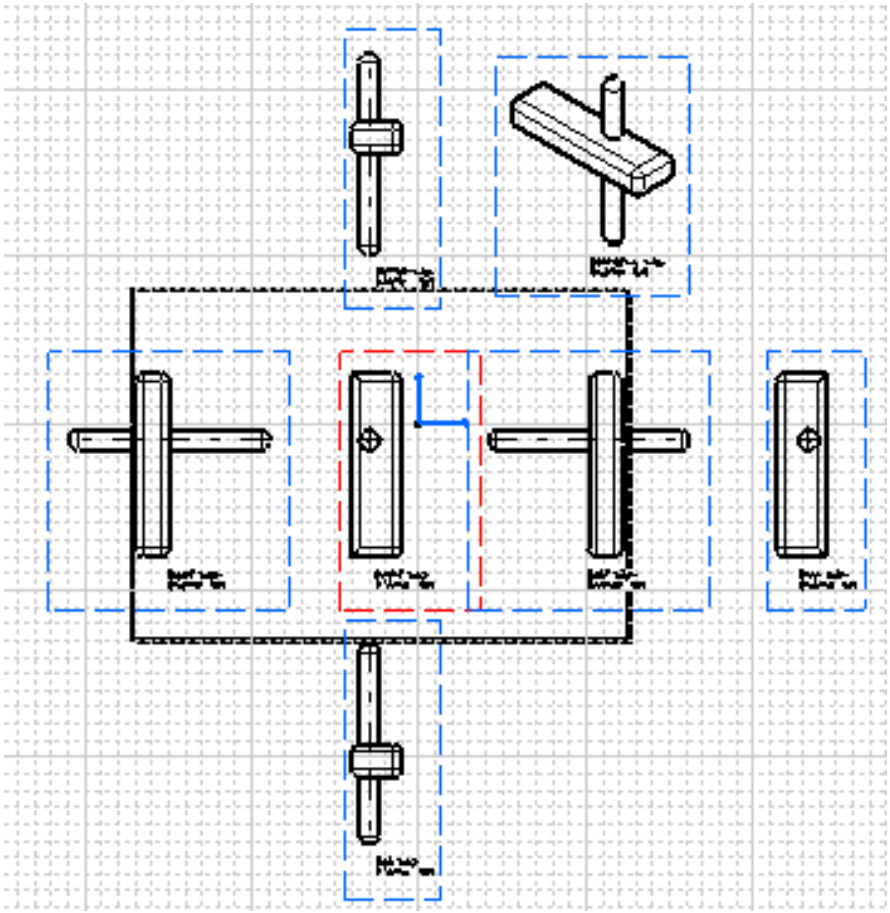
1. Open the PktPadtoInstantiate.CATPart file. The following image displays.



2. From the **Start->Mechanical Design** menu, access the **Drafting** workbench. The **New Drawing Creation** Window displays.

3. Select the **All views** configuration and click **OK**.

4. The drawing corresponding to the pad is generated.



4. Save your drawing and close the file. Click here to see the generated drawing.

5. Go back to the PktPadtoInstantiate.CATPart file to create a part template. To do so, proceed as follows:

   - Select the **Knowledge Templates->Document Template ...** command. The **Document Template Definition** window displays.

   - Click the **Add...** button in the **External documents** field and select the .CATDrawing file you have just created in the **File Selection** window (or use the PktPadDrawing.CATDrawing). Click **Open**.

- Click the **Inputs** tab and select Sketch.1 and Sketch.2 in the geometry or in the specification tree.

- Click the **Published Parameters** tab and click the **Edit List...** button. The **Select parameters to insert** window displays. Select the following parameters using the arrow button:

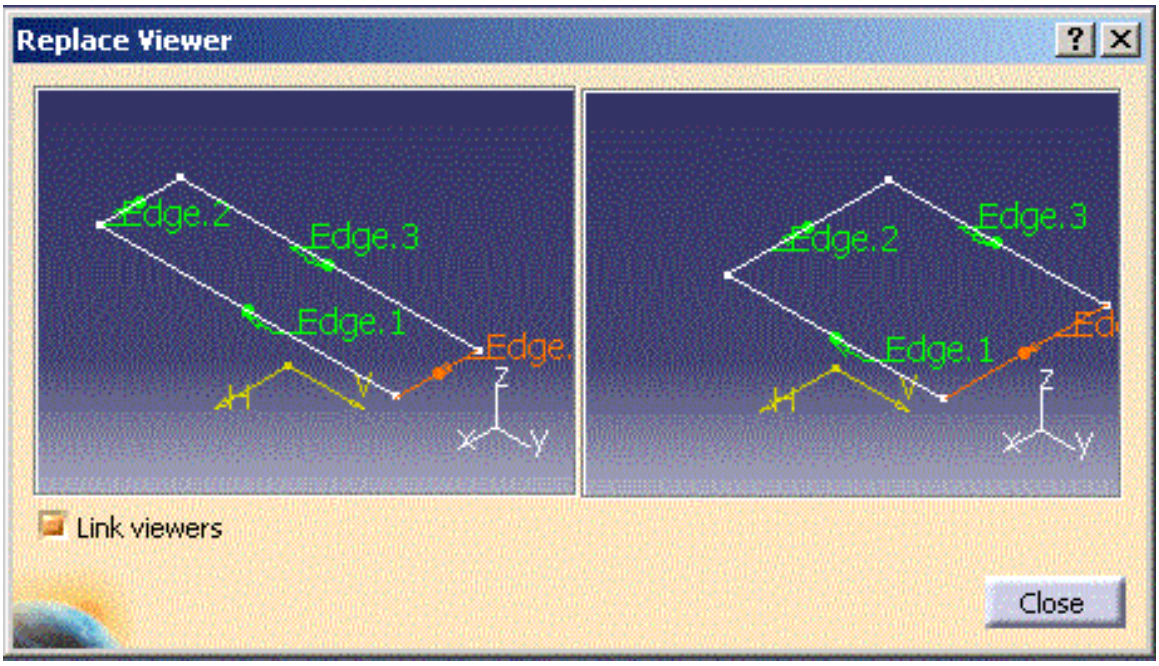
  - PartBody\Pad.1\FirstLimit\Length

  - PartBody\Pad.2\FirstLimit\Length

- In the **Published Parameters** tab, select PartBody\Pad.1\FirstLimit\Length and rename it to Pad_Width in the **Name:** field, then select PartBody\Pad.2\FirstLimit\Length and rename it to Pad_Length.

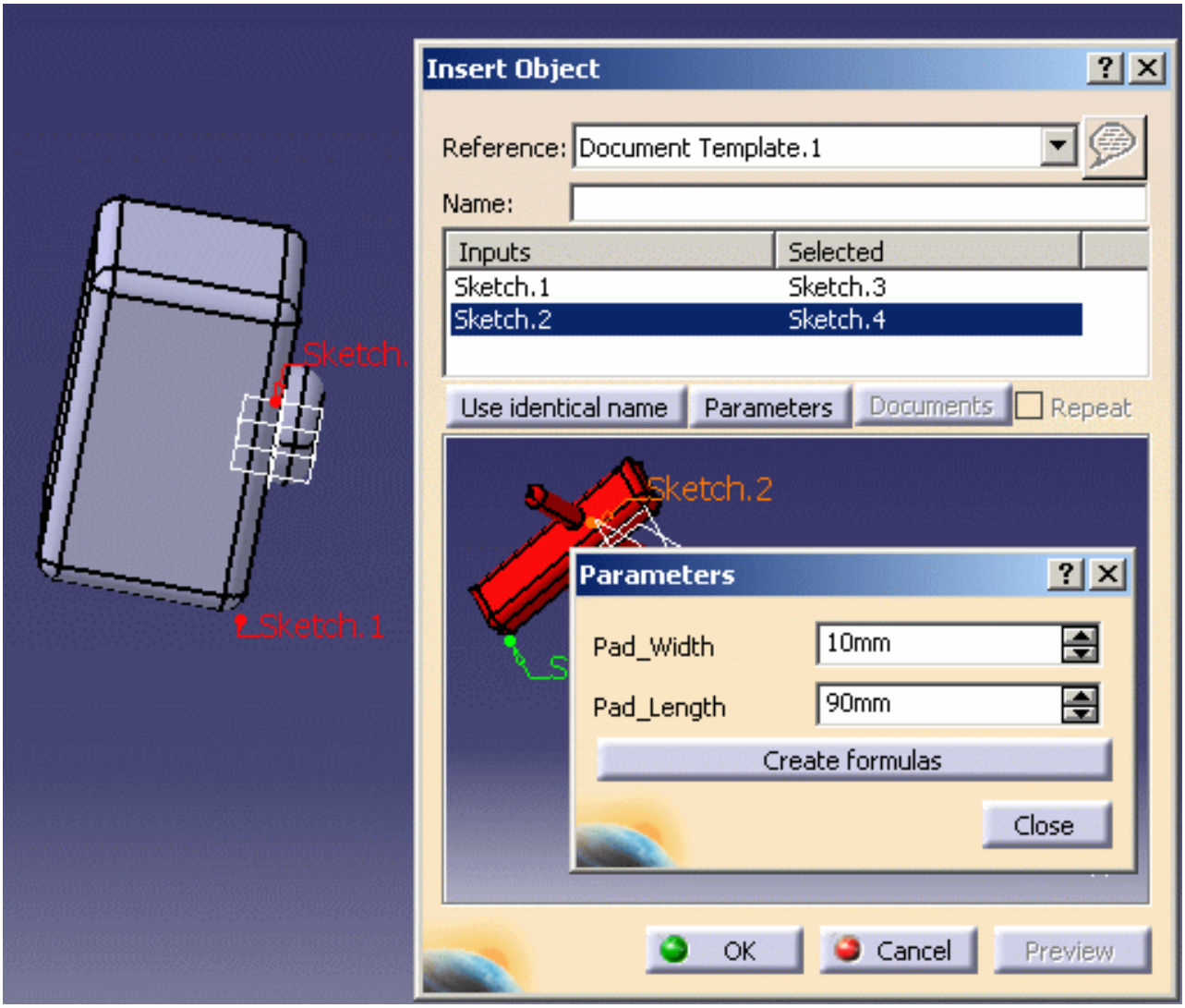
- Click **OK** to validate. Save your file and close it.

6. Open the PktProduct.CATProduct file.
7. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench (if need be).
8. Click the **Instantiate From Document** icon () and select the PktPadtoInstantiate_result.CATPart containing the document template. Click **Open**. The **Insert Object** dialog box displays.
9. Expand the PartBody\Pad.1 node in the specification tree, select Sketch.1, and make the appropriate selections in the opening **Replace Viewer** window (see graphic below). Click **Close** when done.

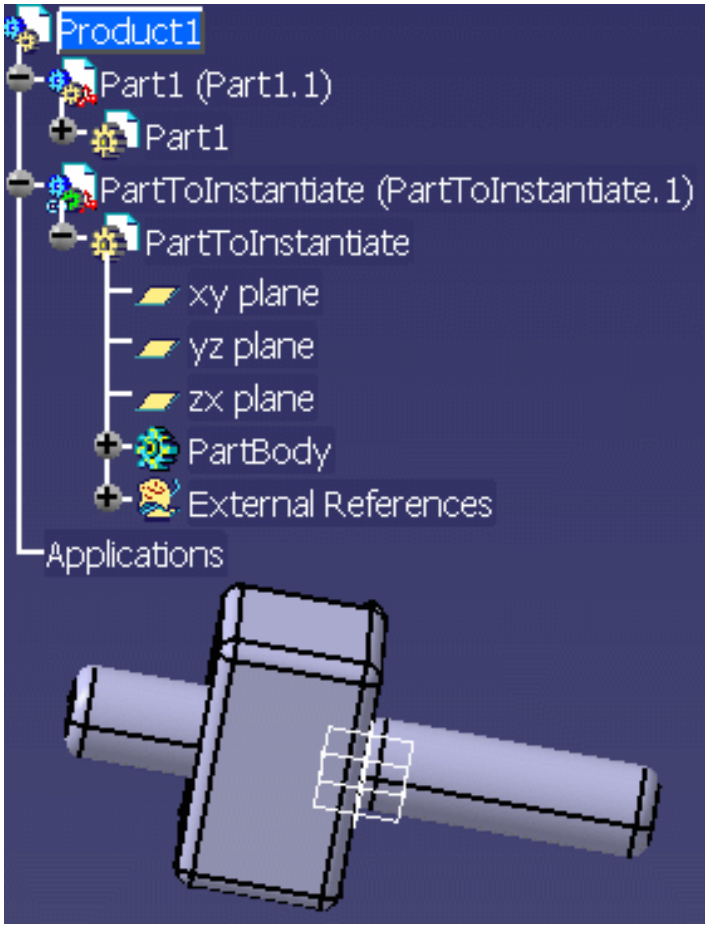


10. Select Sketch.2 in the geometry or in the specification tree.

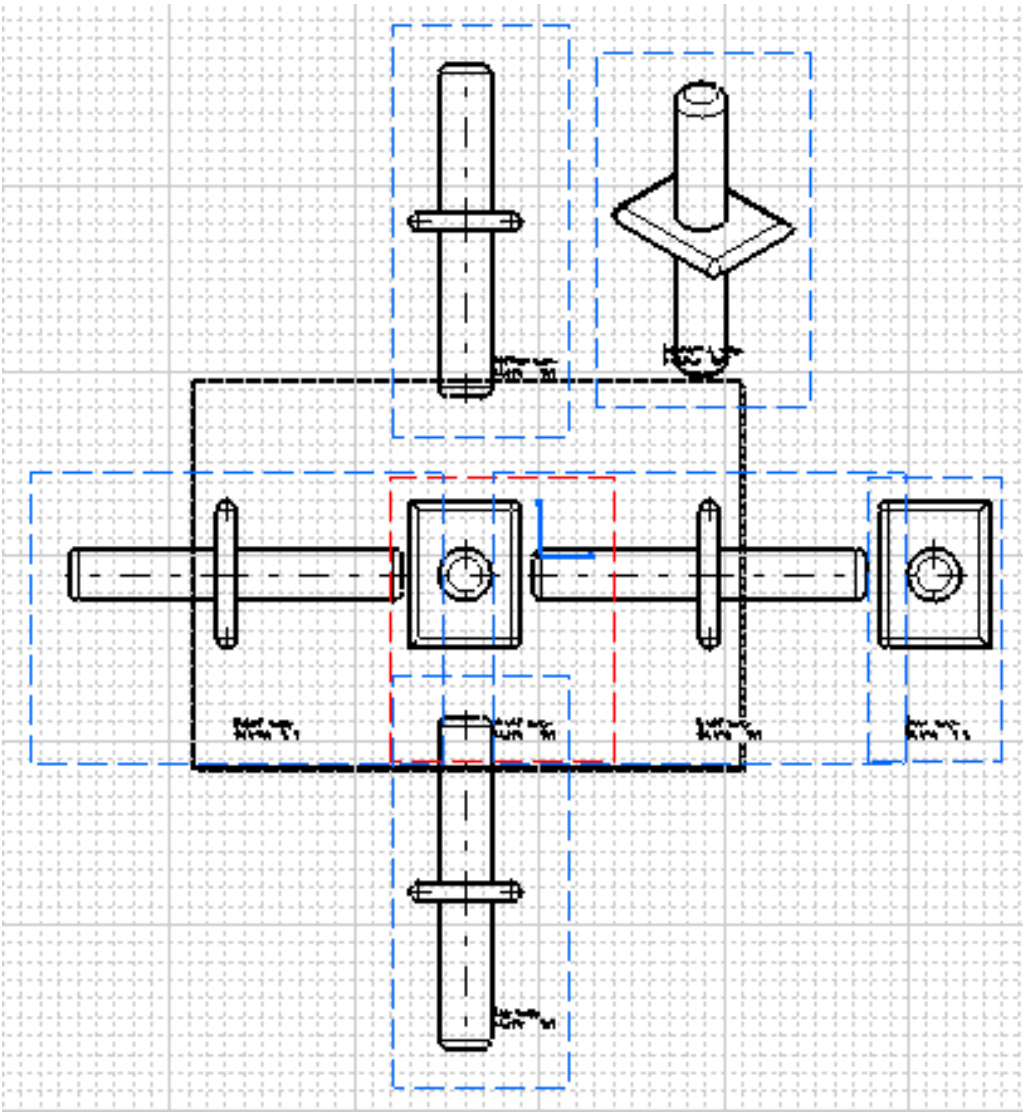
11. Click the **Parameters** button and enter 10mm in the **Pad_Width** field and 90 in the **Pad_Length** field.



12. Click **Close** and **OK** to validate. A message is fired indicating that the external document was regenerated. Click **OK**. The document template was instantiated. (see picture below).



13. From the **Window** menu, access the generated .CATDrawing file. Right-click CATDrawing2 in the left part of the window and select the **Update Selection** command. The drawing is updated and matches the new product.



Refer to the Quick Reference topic for a comprehensive list of the interactions that can be carried out on document templates.

Document Templates: Methodology

- It is possible for the user to define document templates based on contextual products and parts or on isolated parts and products. It is highly recommended to work with isolated documents: not so many documents will be instantiated (when working with contextual products, the context products are needed for instantiation).
- The assembly structure of the documentation template should not be modified after the document template definition (you cannot add or remove documents for example.)

To know more about Part and Assembly Templates...

Part and Assembly Templates are templates that work at the part or at the assembly level.

The **Document Template Definition** window can be accessed by selecting the **Insert->Document Template Creation...** command from the following workbenches:

- Part Design
- Generative Shape Design
- Wireframe and Surface Design
- Assembly Design
- Product Structure

Working with Part Templates

A part created in Catia may contain user parameters and geometry data. It is not a contextual part. The user can create a part template that references that part. This template is a feature that is created in the CATPart document itself (very similar to the PowerCopy definition) and stored in a catalog. Several part templates may be defined in the same CATPart document.

To create a part template, the user:

- selects parameters and geometry data that will be considered as the template inputs (he can assign a role and a comment to each input).
- publishes some internal parameters (name and comment). The part number is automatically published.
- gives a name, comment, URL, icon to this template.

In product structure context, the part is inserted as a component of the current product.

Working with Assembly Templates

A user creates an assembly interactively. Then, he wants to create an assembly template that references the root product of this assembly.

To create an assembly template, the user:

- selects parameters and geometry data that will be considered as the template inputs (he can assign a name to each input).
- publishes some internal parameters (name and comment).
- chooses if:

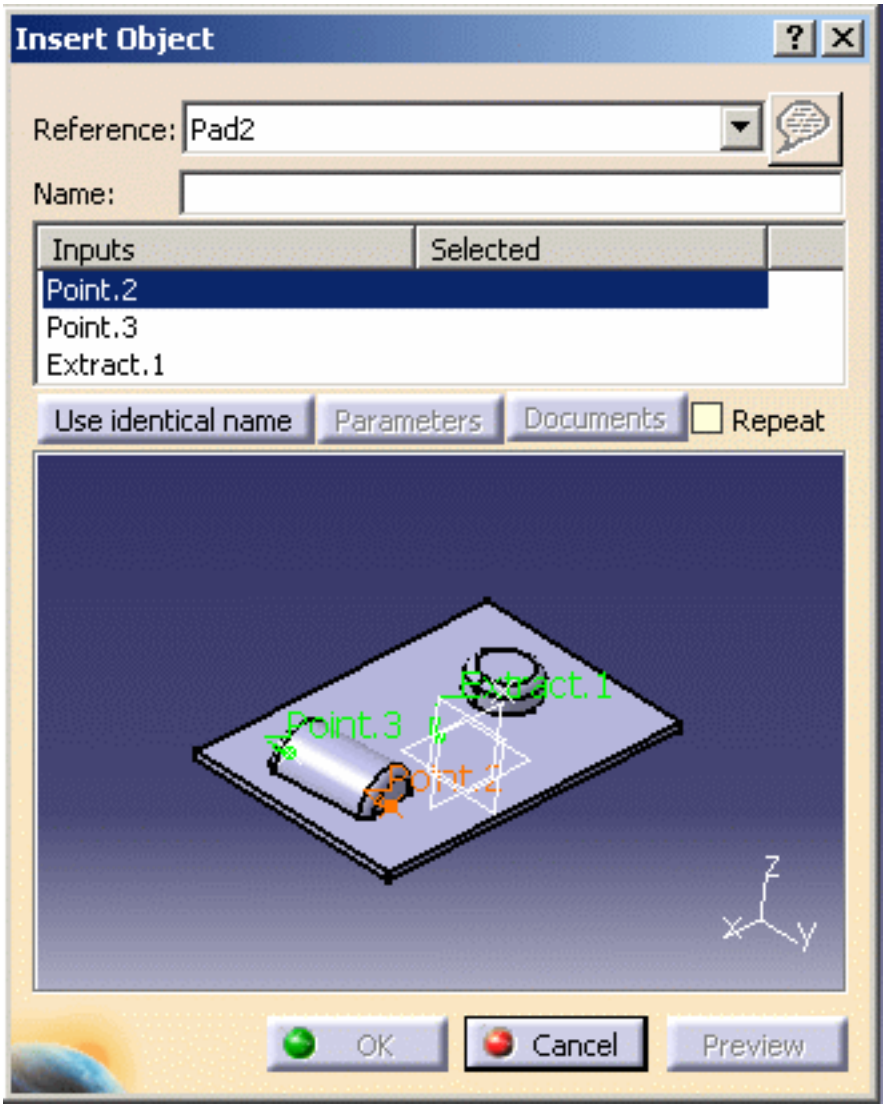
- the part numbers of replicated components are automatically published.
- for each part or each sub-assembly, this sub-component will be replicated at instantiation or if only a reference to this sub-component will be created (a standard component).
- he wants to select external documents (Drawings / Analysis) that references elements of the product structure. Those elements will be replicated at instantiation.

- assigns a name, comment, URL, icon to this template.

The template definition is a feature located in the CATProduct document itself. Several assembly templates may be defined in the same CATProduct document.

To know more about the Insert Object dialog box ...

The **Insert Object** dialog box is displayed when instantiating a PowerCopy, a User Feature or a Document Template.



The Reference scrolling list enables the user to select the feature that he wants to instantiate if many advanced features (PowerCopies, user features and document templates) were defined.

The Comments & URLs icon () is available with user features and Document templates only. It is always grayed out when instantiating Power Copies. If a URL was added to a user feature or a Document template, clicking this icon enables the user to access the URL. To know more about this function, see the *Knowledge Advisor User's Guide*.

The **Name** field enables the user to change the name of the user feature instance.

Use identical name	Features	This function searches in the whole CATPart for features having the name of the input. If a feature with the input name is found, this feature is automatically used as input
	Publications	If a publication with the input name is present in the CATPart, the input will be valuated with the published element. The search will be performed only in CATPart files (not in CATProduct).

Sub-Elements

- If the input is a sub-element of an Axis System (for instance the input name is "Axis System.1\XY Plane") and an axis system named "Axis System.1" is to be found in the CATPart, the "Use Identical Name" function will automatically create a sub-element on Axis System.1.
- If the input has for name "Point.1/Vertex" (resp "Line.1/Edge", "Surface.1/Face") and Point.1 (resp Line.1, Surface.1) is made of only one sub-element, then the input will be valuated automatically.

Parameters

- When an input parameter of a Power Copy or a User Feature belongs to a Geometrical Element or a Parameter Set, in the definition process of the PC or UDF, the default input name of the parameter is computed relatively to its father name. For instance for the parameter X of Point.1, the default input name will be "Point.1/Point Coordinates.1/X". For a parameter in a parameters set, the full name of the parameter will be used because different parameters set can have the same name so we need the full name to identify the right parameter. For instance, for the second parameter Length.1, the default input name will be "Parameters/Parameter.1/Parameters.1/Length.1"
- **Use Identical Name** will look for the parameters published by the geometrical elements. For instance, if the name of the input is "Point.1/Point Coordinates.1/X" (default name) and if a Point.1 is found in the CATPart, the input will be automatically valuated with the parameter X of Point.1.

  It will work the same way with parameters of a parameter set. For instance, if the name of the input is "Parameters.1/Parameters.1/Length.1 (default name) and if a parameter whose absolute path name is "Parameters.1/Parameters.1/Length.1" is found in the CATPart, the input will be automatically valuated with the parameter Length.1 of the Parameters set.
- Multiple solutions Management (V5R11): In case of multiple solutions found for one input name, there will be no automatic valuation and the user will have to choose the desired one by itself.

Parameters This option enables to display the **Parameters** dialog box and modify values if need be. It also enables the user to create formulas by clicking the **Create formulas** button on every parameter with the same name provided there are any.

Documents This option enables the user to access the list of documents (such as design tables) pointed by one of the elements making up the template.
If there are documents, the Documents dialog box opens and you can click the Replace button to display the File Selection dialog box and navigate to a new design table to replace the initial one.
When no document is referenced, the Documents button is grayed out.

Repeat Check the **Repeat** button to repeat the instantiation.
In this case, once you have clicked OK in the Insert Object dialog box, the latter remains open, the template's **Inputs** are listed and ready to be replaced by new inputs, as described above.
Modified parameters using **Parameters** button are retained as well for the next instantiation.
To exit the command, uncheck the **Repeat** button before the last instantiation or click **Cancel**.

Interactive Templates Quick Reference

This topic is intended for those of you who need a quick answer to their questions about the interactive templates. However, using this part requires a prerequisite knowledge of templates as no detail is given.

User Features

Creating a User Feature Scenario

1. Open a .CATPart file.
2. Select the **Insert->UserFeature->UserFeature Creation...** command from the standard menu bar or click the **Create a UserFeature** icon (). The Userfeature Definition dialog box is displayed.
3. Replace the default user feature name, then select the object in the specification tree.
4. Select the Outputs tab. Specify the main result.
5. Click **OK** in the dialog box. The user feature is added to the specification tree.

Saving a User Feature in a Catalog Scenario

1. Open a .CATPart file.
2. Click the **Save in Catalog** icon () from the standard menu bar. The 'Catalog save' dialog box is displayed.
3. Select the Create a new catalog option and click the button on the right-hand side of the Catalog name field. The dialog box displayed allows you to create a **.catalog** file where to store the created user features. Enter a file name and click Open. Then click **OK** in the Catalog save dialog box. The catalog containing the user feature is created.

Instantiating a User Feature from a Catalog Scenario

1. Open a .CATPart file.
2. In the standard toolbar, click the **Open Catalog**  icon. The catalog browser is displayed.
3. Click the  icon. In the dialog box which is displayed, select the catalog which contains the user feature(s) that you want to instantiate. Click Open to open the selected catalog. The dialog box which is displayed next depends on your last interaction on this catalog. Double-click the object displayed in the left pane until the user feature is available.
4. To instantiate the object into the document, proceed as follows:
    a. If need be, select the feature in the Insert Object dialog box, then select the feature in the document geometry area or in the specification tree.

b. Click the Parameters button. The dialog box which is displayed provides you with the way to modify the parameter you have declared as published at the user feature creation. Modify the value (if necessary).

c. Click OK to instantiate the user feature and exit the Insert Object dialog box. The user feature is instantiated into the document.

Instantiating a User Feature from a Document Scenario

1. Open a .CATPart file.
2. Click the Instantiate an element stored in a document icon. The **File Selection** dialog box displays.
3. Select the .CATPart file containing the user feature to instantiate, and click **Open**. The Insert Object dialog box displays.
4. To instantiate the object into the document, proceed as follows:
   a. If need be, select the feature in the Insert Object dialog box, then select the feature in the document geometry area or in the specification tree.
   b. Click the Parameters button. The dialog box which is displayed provides you with the way to modify the parameter you have declared as published (if any) at the user feature creation. Modify the value (if necessary).
   c. Click OK to instantiate the user feature and exit the Insert Object dialog box. The user feature is instantiated into the document.

Instantiating a User Feature from a Selection Scenario

1. Open the CATPart file that will contain the user feature instance as well as the file containing the user feature that you want to instantiate.
2. Tile the window vertically.
3. Expand the KnowledgeTemplates node in the file containing the user feature and click the user feature once.
4. Go to the file that will contain the user feature and click the **Instantiate from Selection** icon (). The **Insert Object** dialog box displays.
5. Make the appropriate selections and click OK to instantiate the user feature and exit the **Insert Object** dialog box. The user feature is instantiated into the document.

Editing a User Feature

To edit a user feature, double-click it in the specification tree to display the **UserFeature Definition** dialog box and edit its content. Note that as far as user features are concerned, only the following actions can be performed in Edit mode:

- Renaming inputs
- Publishing parameters
- Modifying parameters values
- Associating an icon with the user feature
- Modifying the main result
- Creating a type associated with the user feature

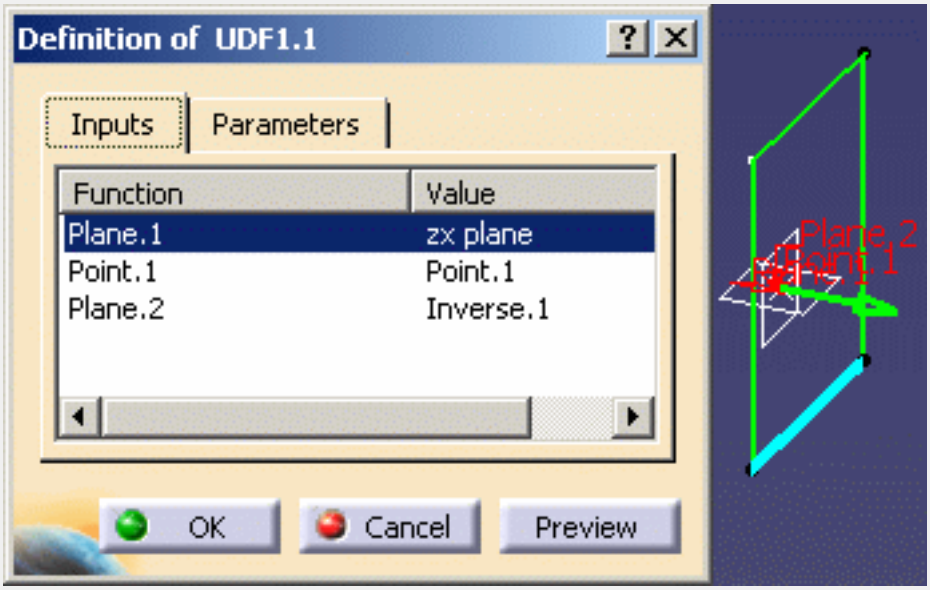
Note that the user feature definition cannot be modified after leaving the Definition tab during the creation process.

Managing the orientation of the items making up the user feature (planes, curves, ...) after instantiating the user feature in Edit mode

1. Open the PktManagingPlanes.CATPart file.
2. From the Standard menu, select the **Insert->Instantiate from Document...** command.
3. In the **File Selection** panel, select the PktPlaneUDF.CATPart file and click **Open**.
4. The **Insert Object** dialog box displays. Select:
   - Plane.1=zx plane
   - Point.1=Point.1
   - Plane.2=xy plane
5. Reverse the direction of Plane.1 and Plane.2 and click **OK** when done. The user feature is instantiated.
6. Double-click UDF.1 in the specification tree. The Definition box of the user feature displays.

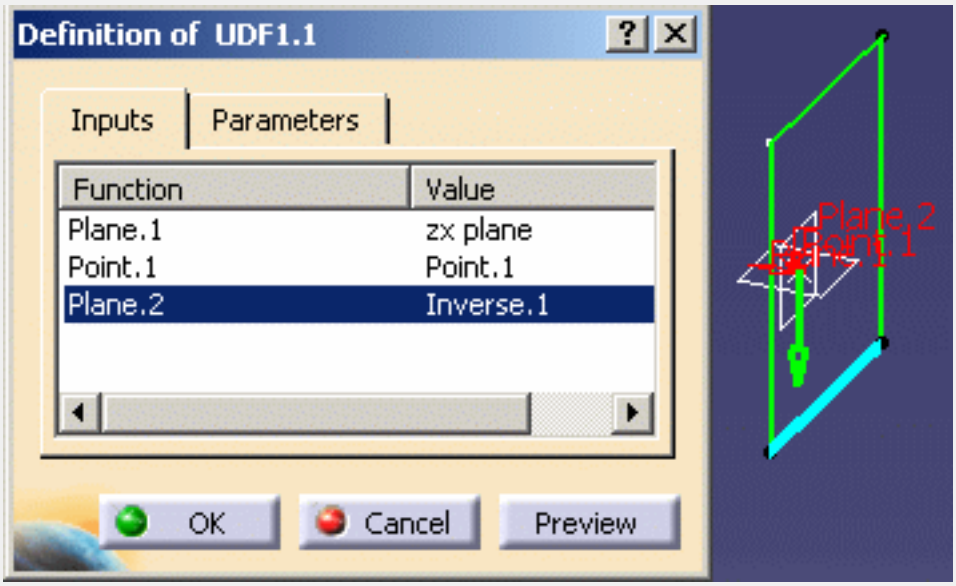
   Note that even if the selected inputs are both planes, in the first case (Plane.1), the display shows the zx plane as selected, and in the second case (Plane.2), the display shows a new element (Inverse.1) which was created and inserted at instantiation. Click **OK** to exit the dialog box.
7. Open the PktPlaneUDF.CATPart file and double-click the user feature:
   - The input corresponding to Plane.1 is pointed by a Sketch: The orientation of Plane.1 is performed inside the Sketch feature. That is why the input is still the zx plane.
   - The input corresponding to Plane.2 is pointed by a surfacic feature (Extremum.1): To manage the orientation of surfacic features, an Inverse feature is created. That is why the input references Inverse.1.
8. Go back to the PktManagingPlanes.CATPart file and double-click the UDF1.1.
9. Select Plane.1 in the Definition box.

- Note that the orientation of the zx plane is not identical to the one selected by the user at instantiation. It is a default orientation for the input. If you click the corresponding arrow, you can modify this orientation to get the desired one.



10. Select Plane.2 in the Definition box.

- Note that the orientation is the default one for the Inverse.1 feature. If you click the corresponding arrow in the geometry, you can modify the orientation of the user feature input. You will see then that the Inverse feature disappears or reappears depending on the direction of the arrow.



Working with the UserFeature Definition Window

Renaming an Input

To rename an input:

- Click the Inputs tab in the Userfeature Definition window.
- Select the input whose name is going to be modified in the graph.
- Change its name in the Name field and click OK when done.

Publishing Parameters

It is possible to publish parameters. This way, when instantiating the user feature, the user can edit these parameters on the user feature instance. Published parameters appear under the user feature reference in the specification tree.
To publish a parameter:

- Click the Parameters tab in the Userfeature Definition window
- Select the parameter intended to be modified in a forthcoming instantiation and check the Published option.

It is recommended to change the name of the published parameters for them to be meaningful to the end user.

Renaming a Parameter

To rename a parameter:

- Click the Parameters tab in the Userfeature Definition window.
- Select the parameter whose name is going to be modified .
- Check the Published check box and enter the name of the parameter.

Modifying a Parameter Value

To modify the value of a parameter:

- Click the Parameters tab in the Userfeature Definition window.
- Select the parameter whose name is going to be modified.
- Check the Published check box, and enter the new parameter value.

Managing Design Tables

Suppose you include a design table in the UserFeature, you will see the document pointed by the Design Table (as in **Edit->Links**). When instantiating or editing the user feature, you will be able to change the document pointed by the internal design table.

Modifying the Main Result

1. Open the PktModifyingMainResult.CATPart file.
2. Double-click UserFeature1 located below the KnowledgeTemplates node. The Userfeature Definition window displays.
3. Click the **Outputs** tab.

Suppose you are only interested in instantiating the CloseSurface object of Assemble.2.

- Select the Main result output.
- Click the Replace button, then select the CloseSurface.2 feature in the specification tree. The CloseSurface.2 feature will be the only object carried forward to the receiving document during the instantiation process (no supporting pad).

Suppose you want to instantiate the Assemble.2 feature as a whole plus one of the circles required to build the Body.2 object (Circle.2 for example).

- Specify Assemble.2 as a main result
- click the Add button, then select the Circle.2 object in the specification tree. The instantiation process will carry forward the Assemble.2 object and the Circle.2 object to the receiving document.

Note that the dimension of the secondary outputs should always be inferior to the Main result.

Assigning a Type to a User Feature Scenario

1. Open a .CATPart file.
2. Select the **Insert->Userfeature->Userfeature Creation...** command from the standard menu bar.
3. In the **Definition** tab, replace the default user feature name (enter Pad1 as a new name for example) then select a feature in the specification tree.
4. In the **Type** tab, enter the name of the instance type: Enter the first part of the type in the first box, the second part in the second box and hit the **Enter** key.
5. Click the **Manage Type** button. Indicate the Super Type and the Package.
6. Click **Create Type**, **Save**, **Close**.

PowerCopies

Creating a PowerCopy Scenario

1. Open a .CATPart file
2. Select the **Insert ->Advanced Replication Tools -> PowerCopy Creation...**menu item or click the **Create a PowerCopy** icon.
3. Select the elements making up the PowerCopy from the specification tree.
4. Define the PowerCopy as you wish to create it.
5. Click OK to create the PowerCopy.

Saving a Power Copy in a Catalog Scenario

1. Open a .CATPart file containing a PowerCopy. The PowerCopy displays below the PowerCopy node.
2. Click the **Save in Catalog** icon () from the standard menu bar in the PKT workbench. The 'Catalog save' dialog box displays.
3. Select the Create a new catalog option and click the button on the right-hand side of the Catalog name field. The dialog box which is displayed allows you to specify a .catalog file where to store the created PowerCopies. Enter a file name and click Open. Then click OK in the Catalog save dialog box.

Instantiating a PowerCopy from a Catalog Scenario

1. Open the .CATPart file that will contain the instantiated PowerCopy.
2. In the standard toolbar, click the **Open Catalog**  icon. The catalog browser is displayed.
3. Click the  icon. In the dialog box which is displayed, select the catalog which contains the PowerCopy that you want to instantiate. Click Open to open the selected catalog.
4. Double-click the object displayed in the left pane until you get the object to be instantiated.
5. Double-click the object. The Insert Object dialog box is displayed.
6. If need be, select the feature in the Insert Object dialog box, then select the feature in the document geometry area or in the specification tree.
7. Click OK to instantiate the PowerCopy and exit the Insert Object dialog box. The PowerCopy is instantiated into the document.

Instantiating a PowerCopy from a Document Scenario

1. Open a .CATPart file.
2. Click the Instantiate an element stored in a document icon () . The File Selection dialog box displays.
3. Select the .CATPart file containing the PowerCopy to instantiate, and click **Open**. The Insert Object dialog box displays.
4. To instantiate the object into the document, proceed as follows:
    a. If need be, select the feature in the Insert Object dialog box, then select the feature in the document geometry area or in the specification tree.
    b. Click the Parameters button. The dialog box which is displayed provides you with the way to modify the parameter you have declared as published (if any) at the PowerCopy creation. Modify the value (if necessary).
    c. Click OK to instantiate the PowerCopy and exit the **Insert Object** dialog box. The PowerCopy is instantiated into the document.

Instantiating a PowerCopy from a Selection Scenario

1. Open the .CATPart file that will contain the PowerCopy as well as the file containing the PowerCopy that you want to instantiate.
2. Tile the window vertically.
3. Expand the PowerCopy node in the file containing the PowerCopy and click the PowerCopy once.
4. Go to the file that will contain the PowerCopy and click the **Instantiate from Selection** icon (). The **Insert Object** dialog box displays.
5. Make the appropriate selections and click OK to instantiate the PowerCopy and exit the **Insert Object** dialog box. The PowerCopy is instantiated into the document.

Editing a PowerCopy

To edit a PowerCopy, double-click it in the specification tree to display the **PowerCopy Definition** dialog box and edit its content.

Introducing the PowerCopy Definition Window

Renaming Inputs

To rename an input:

- Click the Inputs tab in the PowerCopy Definition window.
- Select the input whose name is going to be modified in the graph.
- Change its name in the Name field and click OK when done.

Publishing Parameters

To publish a parameter:

- Click the Parameters tab in the PowerCopy Definition window.
- Select the parameter intended to be modified in a forthcoming instantiation.
- Check the Published option.

Modifying a Parameter Value

To modify the value of a parameter:

- Click the Parameters tab in the PowerCopy Definition window.
- Select the parameter whose name is going to be modified.
- Check the Published check box, and enter the new parameter value.

Part and Assembly Templates

Creating a Part Template
Instantiating a Part Template
Introducing the Document Template Definition Window

- Adding External documents
- Assigning a Role to an Input

Creating a Part Template Scenario

1. Open a .CATPart file.
2. From the **Insert** menu, select the **Document Template Creation ...** command or click the **Create a Document Template** icon ().
3. In the **Document Template Definition** window, click the **Inputs** tab and select the inputs you want to select.
4. In the **Document Template Definition** window, click the **Published Parameters** tab to publish parameters (if need be.)
5. Save the file (note that you can save the document template in a catalog.)

Instantiating a Part Template from a catalog Scenario

1. Open a .CATProduct file.
2. Click the Catalog icon and select the catalog you created when creating the template.
3. Double-click the family and the Document Template.1 template.
4. In the **Insert Object** window, click the **Use Identical Name** button in the Insert Object window. Make the appropriate selections in the viewer when necessary and click **OK** when done.

Using the Document Template Definition Window

Adding External documents

To add external documents:

- In the Documents tab, click the **Add...** button. The File Selection window displays.
- Select the file that will be associated to the template.
- Click **Open**.

Note that external documents can only be files of the following types:

- .CATDrawing
- .CATAnalysis
- .CATProcess

Assigning a Role to an Input

To assign a role to an input:

- Click the Inputs tab in the Document Template Definition window.
- Select the input whose name is going to be modified in the graph.
- Change its name in the Name field and click **OK** when done.

Advanced Tasks

Working with Scripting Templates

- The upward compatibility of the Scripting Language is guaranteed.
- For comprehension purposes, some modifications (listed in the following pages) have been made to the Generative Script language. New objects have been added, as well as new attributes. Some types have been removed.
  To know more about the objects and their new equivalents, see Generative Script Objects.
- The objects available in the browser can now be instantiated.

This new section explains how to use the scripting language to create templates. It is divided into 2 different parts: The first part explains the basics of the scripting tool that you can access by clicking the **Create a Generative Script**  icon in the toolbar. This first section is made up of the following topics:

- Creating a Script
- Starting from a Script Skeleton
- Generating the Result of a Script

The second part presents the script: its structure, its syntax, its objects, the commands that can be of use when writing a script as well as more advanced functions. This second section is made up of the following topics:

- Using the Scripting Language
  - Script Structure
  - Generative Script Objects
  - Object Properties
  - Comments
  - Operators
  - Keywords
  - Variables
  - Limitations
  - Using The Generative Knowledge Commands
- Specifying a Context
- Declaring Input Data
- Reusing Input Data

- Tips and Tricks

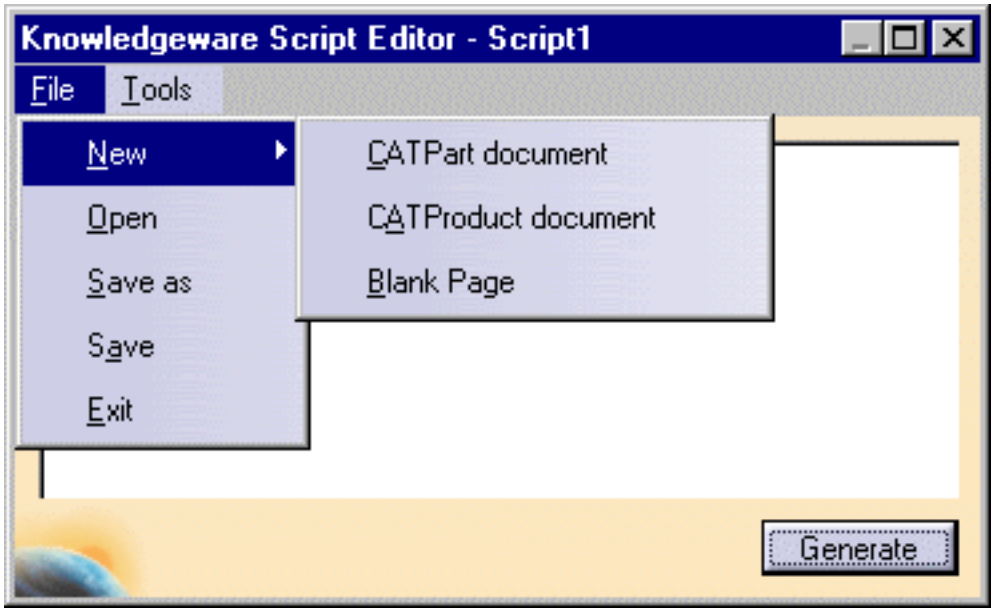
Before creating a loop in a CATPart document, make sure that the **Manual input** option is **unchecked** in the Part Number field of the **Tools->Options->Infrastructure->Product Structure->Product Structure** tab.

Creating a Script

This task explains how to use the script editor as a dialog box to create a script. For information on how to write a script, see Using the Scripting Language

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.

1. Access the Product Knowledge Template workbench by selecting the Product Knowledge Template workbench from the **Start->Knowledgeware** menu.
2. Click the  icon. The Knowledgeware Script Editor is displayed.



3. At this stage, you can create a document skeleton by using the File->New->CATPart or CATProduct document command or by typing the proper instructions into the editor.

   See Using a Script Skeleton for information on how to start from a skeleton.
   See Using the Scripting Language for information on the language.

4. Save your script by using the **File->Save** or **File->Save As** command. Your script is saved in a .CATGScript file.
5. Click **Generate** to create the document.

Starting from a Script Skeleton

This task explains how to write a script by using the Script Editor. This editor provides the user with a way to create a skeleton which reflects the structure of a .CATPart or of a .CATProduct document.

1. In the Product Knowledge Template workbench, click the  icon. The Script Editor is displayed.

2. In the Script Editor, select the **File->New->CATPart** document command. The dialog box is displayed:

3. Fill in the fields and click **OK**. A basic script is created.

```
Mypart isa CATPart
{
        P1 isa Part
        {
                PartBody isa BodyFeature
                {
                }
        }
}
```

4. Save your script and/or generate the related document.

To enrich this script, see Using the Scripting Language.

Generating the Result of a Script

This task shows how to generate a document from a script.

1. Access the Product Knowledge Template workbench by selecting the **Knowledgeware-> Product Knowledge Template** command from the **Start** menu.
2. Click the  icon. The Knowledgeware Script Editor is displayed.
3. Enter your script in the editor or open an already existing .CATGScript file.
4. Click **Generate**. The related document is created in the geometry area.
5. If need be, save your script before exiting the editor.

You can add a new feature to an already existing part provided the name of the part in the script describing this new feature is the same as the name of the part to which you want to add the new feature to. This capability applies to parts generated either from a script or from the Part Design workbench.

Using the Scripting Language

Introducing the Scripting Language

The Scripting Language is a declarative way of generating V5 Features.

What does it allow users to do?

They can describe objects using a very simple script language.

- 3D geometric features (sketches, parts, ...).
- Parameters on features including *formulas*.
- Related positioning & orientation constraints.

They can interactively generate the corresponding V5 models.

How can users use it?

- They can launch the Script editor (graphic mode) in the Product Knowledge Template workbench, open a script file or type it in, then generate the result.
- They can launch a script file in a Rule action using the GenerateScript function.
- They can launch a script stored in a catalog.

Introducing the Scripting Language main Features

The Scripting language enables users to:

- Import definitions from other models/scripts and then instantiate the imported components.
  See Import keyword.
- Incrementally define objects, their properties and the other features they own.
- Define input parameters that will be valued by the user at the beginning of the generation.
  See Declaring Input Data.
- Easily capture generic naming to instantiate contextual features, constraints, ...
  See Using the Generative Knowledge Commands.

Script Structure

A generative script is written in text format and is organized in blocks consisting of related sets of statements. A block consists of 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. A generative script reflects this object hierarchy. In the outermost statement block, you must create the document intended to contain all the features to be created later on.

Example 1

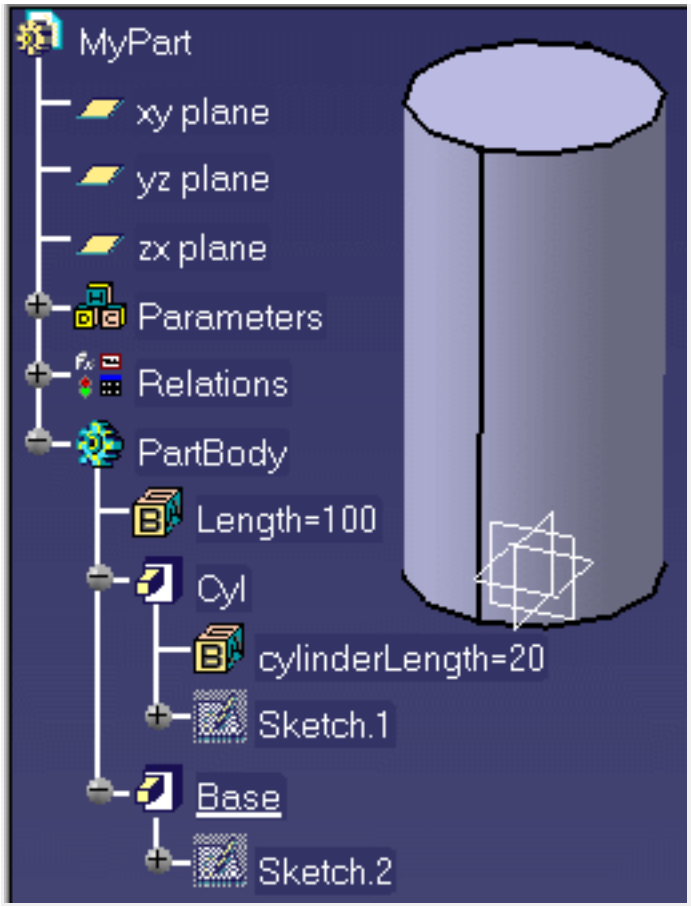
Please find below the basic structure of a script. You can create this skeleton by using the **File->New->CATPart** document command of the Script Editor:

```
part isa CATPart
{
        Mypart isa Part
        {
            PartBody isa BodyFeature
                {
                }
        }
}
```

Example 2

```
MyDocument isa CATPart   // Creates a CATPart
document
{
   MyPart isa Part
   {
      L = 30.0;
      PartBody isa BodyFeature  // Creates the main body
      {
         Length = 100.0; // Length is a parameter of the
part
         Cyl isa Cylinder
        {
          cylinderLength = 20.0;
         }
         Base isa Cylinder // Base is an instance of
Cylinder owned by MyPart
         {
            Height = ?Length * 2; // a formula property
using MyPart/Length;
         }
      }
```

```
    }
}
```



To know more about the isa keyword and the ? operator, see isa keyword and ? operator.

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 SI units.
- When defining properties, the semicolon (;) is a terminator (see example below).

Example

```
myLineDocument isa CATPart
{
      myPart isa Part
      {
            OBody isa OpenBodyFeature
            {
                  Po1 isa GSMPoint
                  {
                        PointType = 0;
                        TypeObject isa GSMPointCoord
                        {
                              X = 50mm;
                              Y = 100mm;
                              Z = 150mm;
                        }
                  }
                  Po2 isa GSMPoint
                  {
                        PointType = 0;
                        TypeObject isa GSMPointCoord
                        {
                              X = 50mm;
                              Y = 0mm;
                              Z = 150mm;
                        }
                  }
                  L isa GSMLine
                  {
                        LineType = 0;
                        TypeObject isa GSMLinePtPt
                        {
                              FirstPoint = object: ../../Po1;
                              SecondPoint = object : ../../Po2;
                        }
                  }
                  L2 isa GSMLine
                        {
                              RefPoint = object: ../../Po1;
                              Values["Start"] = 2 mm;
                              Values["End"] = 20 mm;
```

```
                        RefSkin = object : ../../../`xy-plane`;
                    }
                }
            }
        }
}
```

To know more about the objects, see Generative Script Objects.

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 ;
      }
```

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.

(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.

Keywords

isa	import
from	context
publish	input
in	let

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.
- Required whenever you need to create a feature from a sketch (the script language does not allow you to specify a sketch).

Syntax

import *FileName* **;**

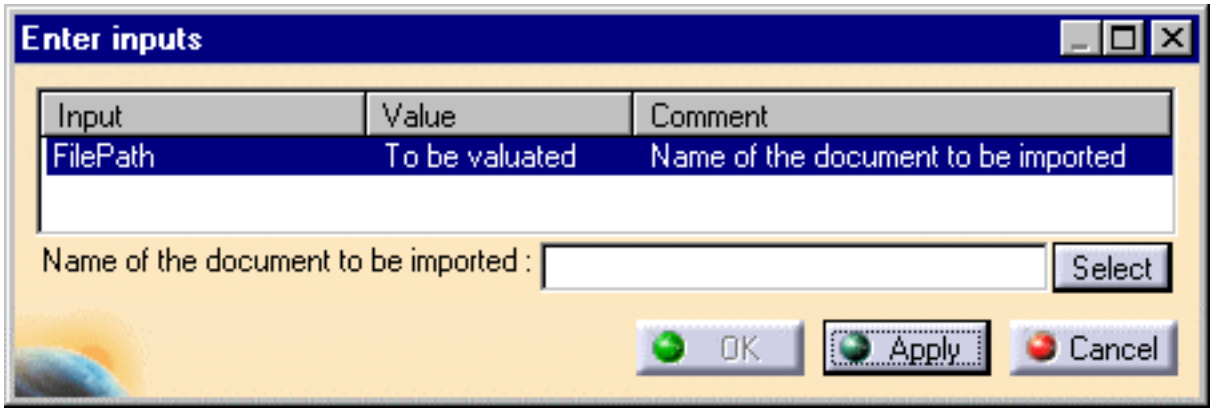
where FileName is the name of the file which contains the document to be imported.
You should enclose the document name within quotation marks and end the import statement with a semicolon (;).

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 or
- Use the Input keyword. A dialog box is displayed when the document is generated. You have to enter the input data required one-by-one to execute the script. For more information, see Input and In Keywords.

Example

The following statement
import Input : FilePath "*Name of the document to be imported*";
displays the dialog box below when the script is executed:



Clicking Select displays a file selection panel. You have to click Apply to make the OK button active.

let Keyword

Definition

Assigns the value of an Input expression to a variable. Using this keyword prevents you from re-entering the value of an input data.

Syntax

let *name* = **Input** : *type_of_input* **;**

Example

```
context Input : Feature "Context ?"


let X = Input : Feature "First Point to reuse";
let Y = Input : Feature "Second Point to reuse";


body isa OpenBodyFeature
{
L isa GSMLine
{
LineType = 0;
TypeObject isa GSMLinePtPt
{
FirstPoint = object: Input X;
SecondPoint = object: Input Y;
}
}
}
```

In Keyword

Definition

Enables the user to create a multiple value parameter. It can be used together with the Input keyword to specify that a piece of data is a multiple value one.

Syntax

DataName = *DefaultValue*, Input: *DataType* In: '*value1*,*value2*,...';

where:

- *DataName* is the name of the piece of data whose value is to be entered by the end-user.
- *DefaultValue* is the default value to be displayed in the "Enter Inputs" dialog box.
- *DataType* is the type of the piece of data whose value is to be specified (see above).

Example

```
myDocument isa CATPart
{
      myPart isa Part
      {
            PartBody isa BodyFeature
            {
                  WWW = 3.6 mm , Input: Length;
                  X = 6.6 mm , Input: Length In : `6.6mm,12 mm,100mm,3.3m` ;
                  Y = 5 , In : `5,10,15,25,50,3304324324`;
                  XX = "relation" , In : `relation1, relation2, relation3`;
                  ZZ = 3, Hide: true;
            }
      }
}
```

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 1

```
Product_Root isa CATProduct
{
Product_Assemblage isa Product
{
Part_For_Publish isa Product
{
Boite isa Part
{
PartBody isa BodyFeature
{
My_Boite isa Pad {}
}
}
publish "!Selection_RSur:(Face:(Brp:(My_Boite;0:(Brp:(Sketch.1;2)));None:());My_Boite)" as
Surf_Normal_To_X_axes;
}
}
}
```

Input Keyword

Definition

Enables the user to declare a value to be entered by the end-user. Before the document is generated, a dialog box is displayed and the user is required to enter a value for each piece of data declared as an input.

The Input keyword may be used in two different contexts:

- If you want a user input to be required, use the following syntax:
  Input:type_of_the_needed_input
  Example: Input:Feature or Input:Length or Input:FilePath

  A user input may be required in the following cases:

  - To define the context of the script: context Input:Feature "Specify the context"
  - To valuate an attribute: X = 10, Input:Length "Enter the value"; or RefPlane = object: Input:Feature "Specify the reference plane";
  - To valuate a name: PartBody, Input:Name "Enter the body name" isa BodyFeature
  - To define an import: import Input : FilePath "Enter the file name";
  - To valuate a local variable: let X = Input : Feature "First Point to use";

- If you want to re-use a local variable.

  Example:

```
let X = Input : Feature "First Point to reuse";
let Y = Input : Feature "Second Point to reuse";
....
L isa GSMLine
{
          LineType = 0;
          TypeObject isa GSMLinePtPt
          {
                    FirstPoint = object: Input X; //X and Y are local
                    SecondPoint = object: Input Y; // variables


     }

} ...

//
```

Here is the list of data that can be declared as an input:

Data	*type*
parameter	parameter type as declared in *f(x)*
file paths	FilePath
feature names	Name
edges when they are used to create features like chamfers or fillets	Edge
points when they are used to create holes	Point
features	Feature
faces	Face
axes	Axis
length	Length

Syntax

DataName = *DefaultValue*, Input: *DataType* "*Comment*"

or

DataName = *DefaultValue*, Input: *DataType* In: '*value1*,*value2*,...';

where:

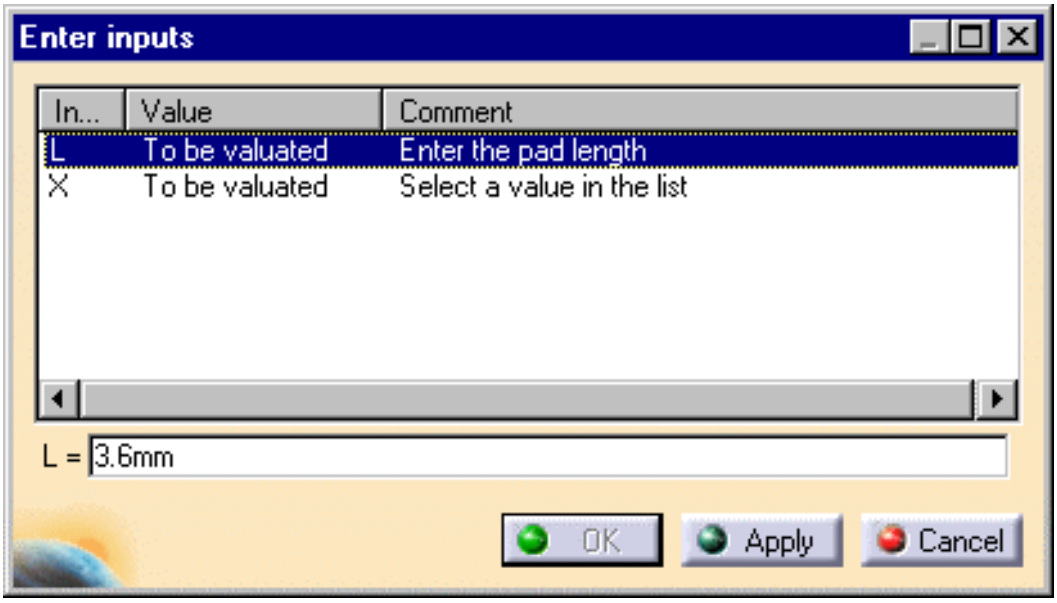
- *DataName* is the name of the data whose value is to be entered by the end-user.
- *DefaultValue* is the default value to be displayed in the "Enter Inputs" dialog box.
- *DataType* is the type of the data whose value is to be specified (see above).
- *valuei* is one of the values of a multiple value data.

Example

When the script below is executed:

```
myDocument isa CATPart
 {
  myPart isa Part
   {
    PartBody isa BodyFeature
     {
      L = 3.6 mm , Input: Length "Enter the pad length";
      X = 6.6 mm , Input: Length "Select a value in the list" In : `6.6mm,12 mm,100mm,3.3m` ;
      FeatureName = "Part1" , In : `Part1,Part2,Part3`;
     }
   }
 }
```

The following dialog box is displayed:

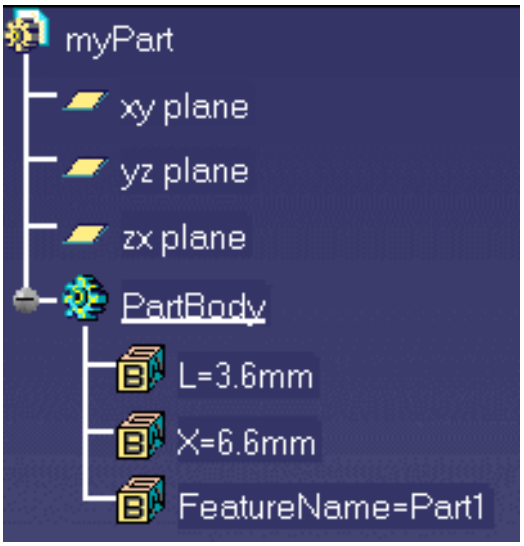


- The L and X values are to be entered by the user.
- The L default value is set to 3.6mm. If you wish to modify this value, select the L line, modify the value in the 'L=' field then click Apply. The X line is then highlighted and the 'X=' field displays a four-value list.

Select one of these values, click Apply then click OK to execute the script and generate the document.

The FeatureName parameter which is a multiple value parameter is created in the specification tree.

You can edit it to check the list of possible values.



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 or
- Use the Input keyword: `h isa CATPart from Input : FilePath "Enter the file path ?"`

Example

```
h isa CATPart from "c:\temp\CATPart"
  {
    h isa Part
      {
      // Additional features
      }
 }
```

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 four different cases:

- It can be used together with the Input keyword. In this case, the user is prompted to enter his inputs.

```
context Input: FilePath
Part isa Part
{
 BBB isa BodyFeature
    {
     Mycylinder isa Cylinder { }
    }
}
```

- 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 { }
```

- It can be used as an argument when the script is generated with a knowledge Expert rule.

```
`$context$`
```

Syntax

- **c**ontext Input: FilePath

or

- context "Mypart.CATPart"

or

- `context `My.CATPart\MyPart\PartBody``

or

- `` `$context$` ``

Example

```
context Input: FilePath
Part isa Part
{
  BBB isa BodyFeature
  {
    Mycylinder isa Cylinder { }
  }
}

context "Part.CATPart"
Mypart isa Part { }

context `Part.CATPart\Mypart\PartBody`
CC isa Cylinder { }
```

isa Keyword

Definition

Creates a typed object or instantiates 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

In the example below, S0 is an instance of the Sketch.0 feature which is imported from the GPS.CATPart document.

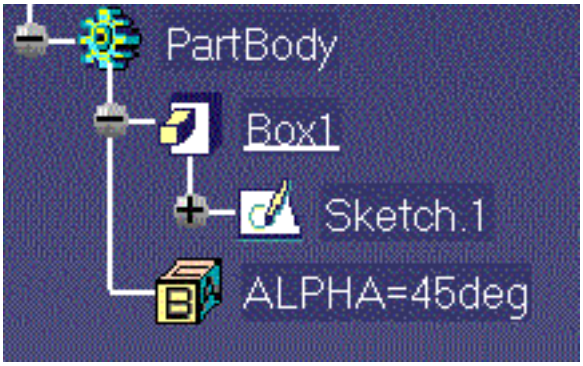
```
import "E:\GPS.CATPart";
myGps isa CATPart
  {
   myPart isa Part
     {
      PartB isa BodyFeature
        {
         S0 isa Sketch.0 {}
         pad0 isa Pad("S0")
         ...
        }
     }
  }
```

The name of the created object should be different from the object type (Box isa Box is incorrect).

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's scope is not really determined by where you declare it. From anywhere in your 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.

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 script.

Generative Script Objects

To display the list of the objects that can be instantiated in PKT, use the Object Browser available through the **Tools->Object Browser...** command.
To know more about the Object Browser, see Using the Object Browser.

Using The Get... Commands

The commands described in this section are the ones the user can access when using scripting language Editor and right-clicking in the Editor window.

When writing a script 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.

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. Access the Product Knowledge workbench, and open the Script Editor.
2. Enter the following script and click **Generate**. 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, 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:()))"){}
            }
       }

}
```

4. Click the **Generate** 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. Access the Product Knowledge Template workbench, and open the Script Editor.
2. Enter the following script and click **Generate**. 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, 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 PktGetSurface.CATPart file.
2. Access the Product Knowledge Template workbench, and open the Script Editor. Enter the following script:

```
import "f:\cube.CATPart";
myFaceDocument isa CATPart
{
      myPart isa Part
      {
            PartBody isa BodyFeature
            {
            P isa Pad{}
            S isa Sketch.1()
```

3. 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.
4. 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 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 PktGetFeature.CATPart file.
2. Access the Product Knowledge Template workbench, and click the Create a Generative Script icon.
3. In the editor, select the File->Open command, and select the PktGetFeature.CATGScript file.
4. Position the cursor after FirstPoint = object: and select the Get Feature command in the contextual menu.
5. Select a point in the geometry and add a semi-colon (;) at the end of the line.
6. Position the cursor after SecondPoint = object: and select the Get Feature command in the contextual menu.
7. Select 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:

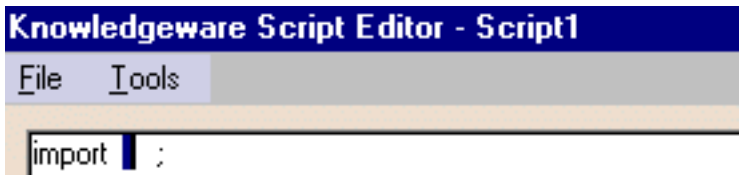
```
Part_OpenBody_Tree isa CATPart
{
Part_OpenBody_Tree isa Part
{
Result_Body isa OpenBodyFeature
{
        Line_Pt_Pt  isa GSMLine
        {
                LineType = 0;
                TypeObject isa GSMLinePtPt
                {
                        FirstPoint = object: Open_body.1\Open_body.2\GSMPoint.3 ;
                        SecondPoint = object :Open_body.1\Open_body.2\GSMPoint.6
                }
        }
}
}
}
```

8. Click the **Generate** button. The line is created.

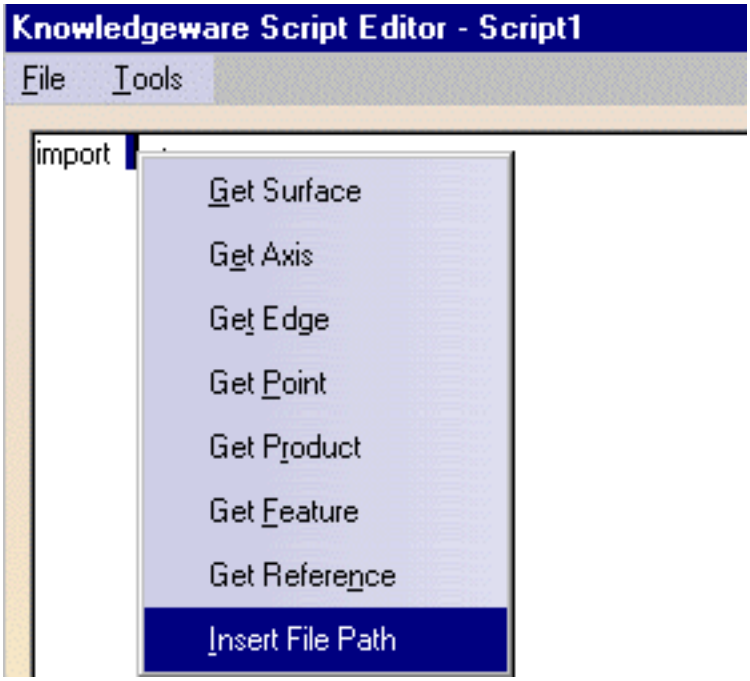
The "Insert File Path" Command

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.

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.

Specifying a Context

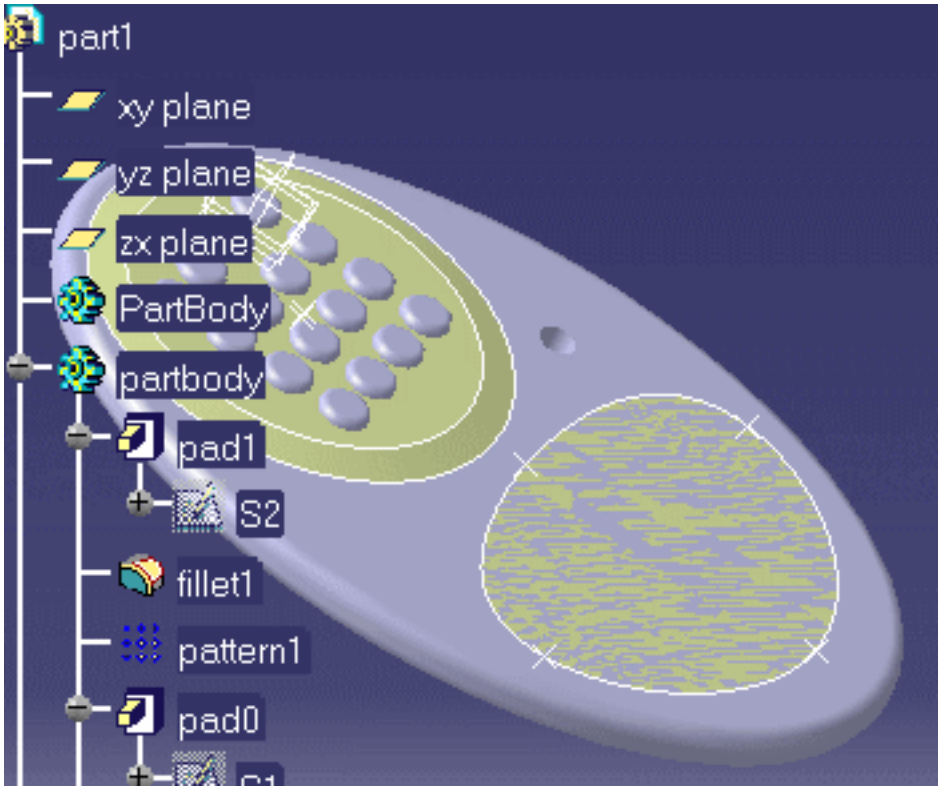
After a document has been generated, you can add new features to this document by executing another script provided you declare in this other script the document you want to add new features to. This declaration is made by using the context keyword. To know more about this keyword, see context keyword.

1. Re-run The Pocket Calculator scenario.
2. Enter the script below into the editor (close the editor from before and reopen a new window or delete all the instructions in the current editor).

```
context `calc.CATPart/part1/partbody`
h1 isa Hole (     )
{ Diameter = 4mm; }
```

   The path specified in the context statement must be surrounded by back quotes.

3. Position the cursor between the parentheses right after the Hole statement, select the Get Feature command from the contextual menu then select a point on the calculator (see the figure below for the location of that point). Then click '**Generate**'. This is what you get on screen.



The h1 feature is added to the document.

Declaring Input Data

This task explains how to declare data as to-be-entered by the user and how to fill in the dialog panel which is displayed before the document is generated. Not all data can be declared as inputs.

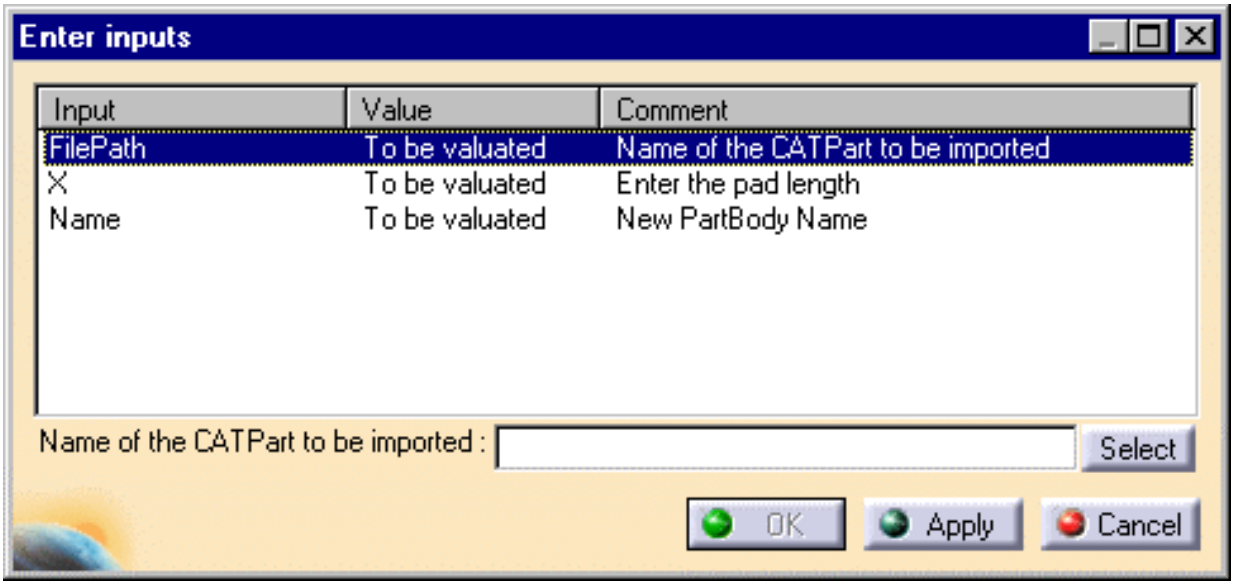
Data can be entered by the user provided they have been declared as Inputs by using the Input keyword in the script. When the script is generated, a dialog box displays the list of values to be entered in order to generate the document. See the Input Keyword for more information on this keyword.

File Paths, Feature Names and Parameter Values

1. In the Product Knowledge Template workbench, click the  icon. The Knowledgeware Script Editor is displayed.
2. Enter the script below

```
import Input : FilePath "Name of the CATPart to be imported";
Pad1 isa CATPart
    {
     mypad isa Part
      {
       X = 10mm, Input : Length "Enter the pad length";
       PartBody, Input:Name "New PartBody Name" isa BodyFeature
        {
         S isa Sketch.1{}
         P isa Pad("S"){}
        }
      }
    }
```

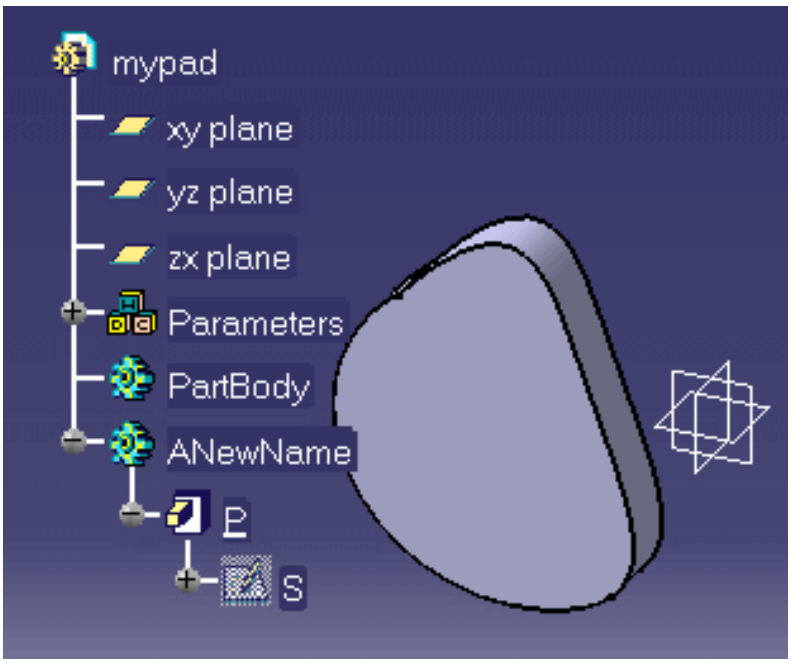
3. Click **Generate**. The dialog box below is displayed:



4. To enter the FilePath input, click Select then select the PktSketchToImport.CATPart sample. Click **Apply**.

5. The X value is now highlighted in the dialog box and you are prompted to enter the X value, that is the pad length. To use the default value, just click **Apply**. Otherwise, enter a new value in the X field, then click **Apply**.

6. If need be, repeat this operation for the Name input. Click Apply to enter the Name value. The **OK** button should now be active. (The **OK** button is grayed out as long as there is still one or more inputs to be specified). Click **OK**. The extruded pad below is generated:



Edges

1. In the Product Knowledge Template workbench, click the  icon. The Knowledgeware Script Editor is displayed.

2. Enter the script below:

```
MyBox isa CATPart
 {
  BoxPart isa Part
   {
   PartBody isa BodyFeature
    {
      Box1 isa Box
      {
        Width = 20.0 mm ;
        Height = 25.0 mm ;
        Length = 15.0 mm ;
       }
    }
   }
 }
```

3. Click **Generate**. A box is displayed in the geometry area.

4. Back to the script editor. Add the statements related to the fillet and hole creations to your script:

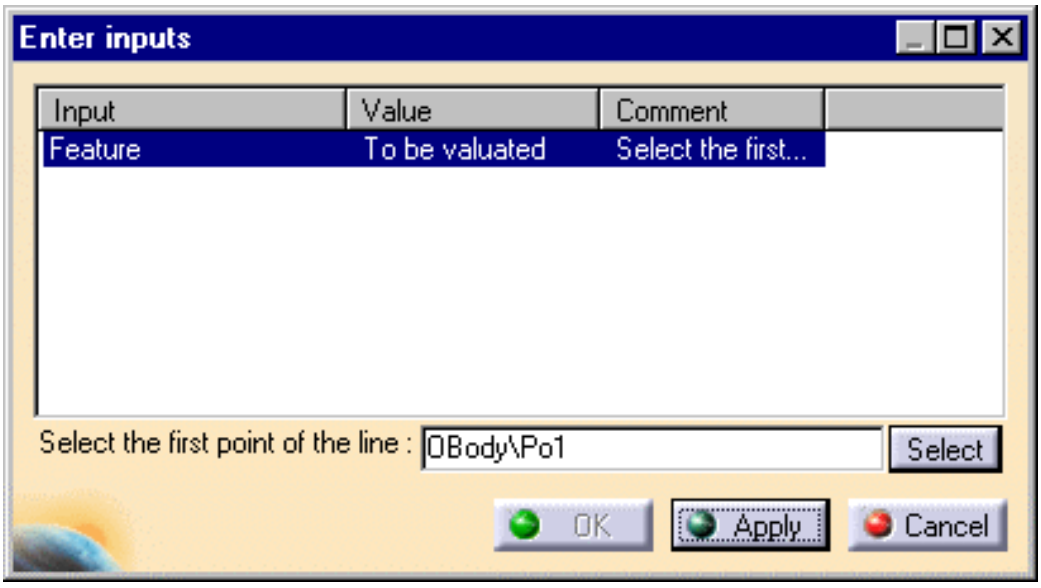
```
MyBox isa CATPart
 {
  BoxPart isa Part
   {
    PartBody isa BodyFeature
     {
      Box1 isa Box
       {
        Width = 20.0 mm ;
        Height = 25.0 mm ;
        Length = 15.0 mm ;
       }
       /* Added statements - Start */
       fillet2 isa ConstantEdgeFillet (fillet2 isa ConstantEdgeFillet
       ("Edge:(Face:(Brp:(Pad.1;0:(Brp:(Sketch.1;3)));None:();Cf9:());Face:(Brp:
       (Pad.1;2);None:();Cf9:());None:(Limits1:();Limits2:());Cf9:())"))
              {
               Radius =2.0 mm;
              }
       /* Added statements - End */
    }
  }
```

```
}
```

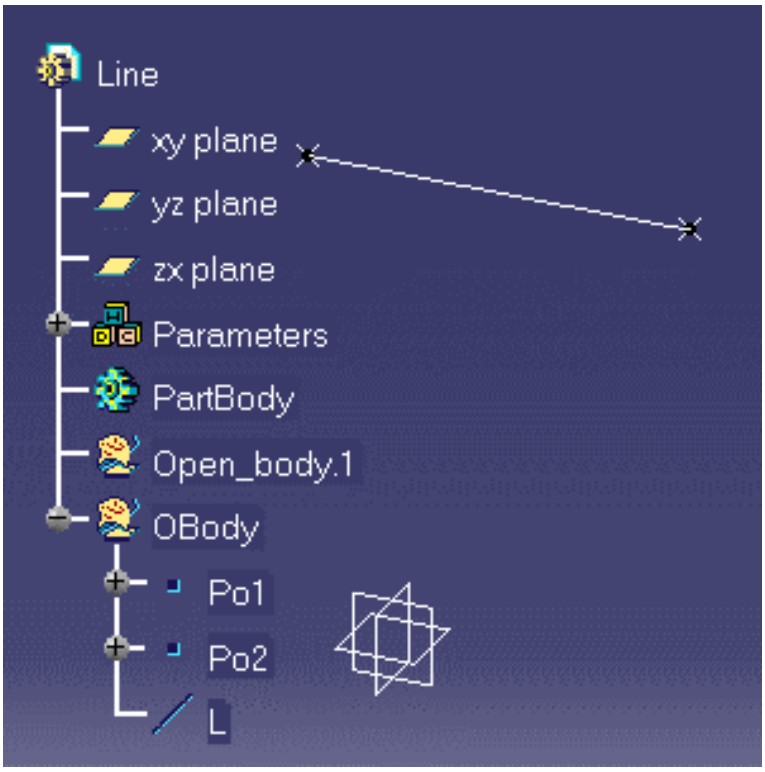
5. Click **Generate**. The fillet is created.

Features

1. In the Product Knowledge Template workbench, click the  icon. The Knowledgeware Script Editor is displayed.
2. Use the **File->Open** command to open the PktInputFeature0.CATGscript macro which is delivered as a sample. Don't remove the comments corresponding to the line creation.
3. Click Generate. The Po1 and Po2 points are created in the geometry area and the specification tree is updated accordingly.
4. In the script editor, remove the comments in order to create L, then click Generate. The "Enter Inputs" dialog box is displayed. Only one input is to be entered by the end-user. Click **Select**, then select the Po1 feature in the specification tree.



5. Click **Apply** then **OK**. The line joining Po1 and Po2 is created.



Reusing Input Data

The scenario below explains how to reuse input data using the let keyword. To know more about this keyword, see let Keyword.

1. Re-run The Pocket Calculator scenario. Keep the generated document open.
2. Enter the script below in the editor:

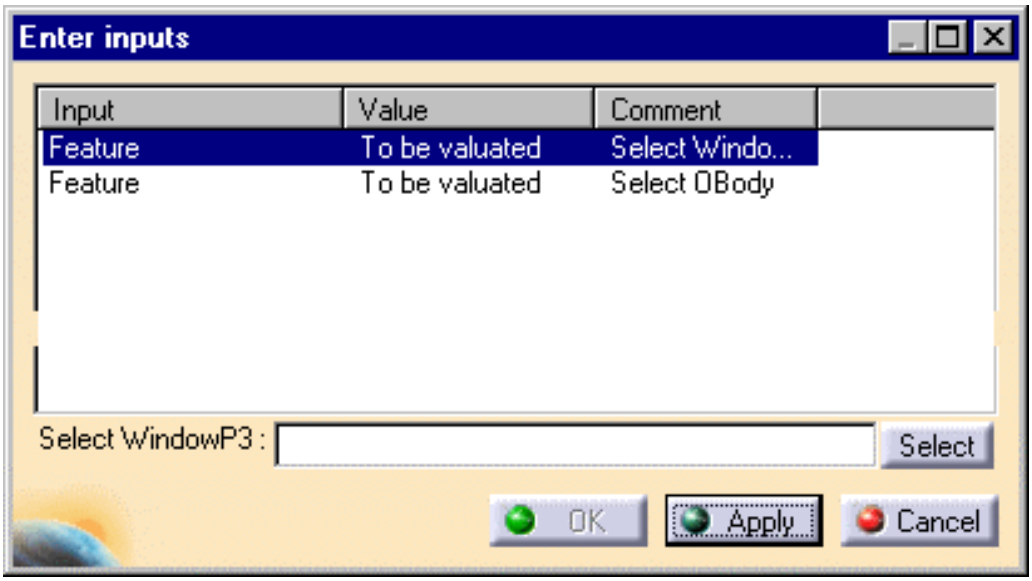
```
context Input : Feature "Select OBody"
P1 isa GSMPoint
  {
   PointType = 0;
   TypeObject isa GSMPointCoord
    {
     X = 50mm;
     Y = 0mm;
     Z = 150mm;
    }
  }
P2 isa GSMPoint
  {
   PointType = 0;
   TypeObject isa GSMPointCoord
    {
     X = 50mm;
     Y = 100mm;
     Z = 150mm;
    }
  }

Line1 isa GSMLine
  {
   LineType= 0 ;
   TypeObject isa GSMLinePtPt
    {
     FirstPoint = object : Input : Feature "Select WindowP3";
     SecondPoint = object: ../../P1;
    }
   }

Line2 isa GSMLine
   {
    LineType= 0 ;
    TypeObject isa GSMLinePtPt
     {
      FirstPoint = object : Input : Feature "Select WindowP3";
      SecondPoint = object: ../../P2;
     }
    }
```

3. Click **Generate**. The **Enter Inputs** dialog box is displayed. You are prompted to enter the OBody feature, then twice the WindowP3 feature. Click **Cancel** and proceed to the next step.

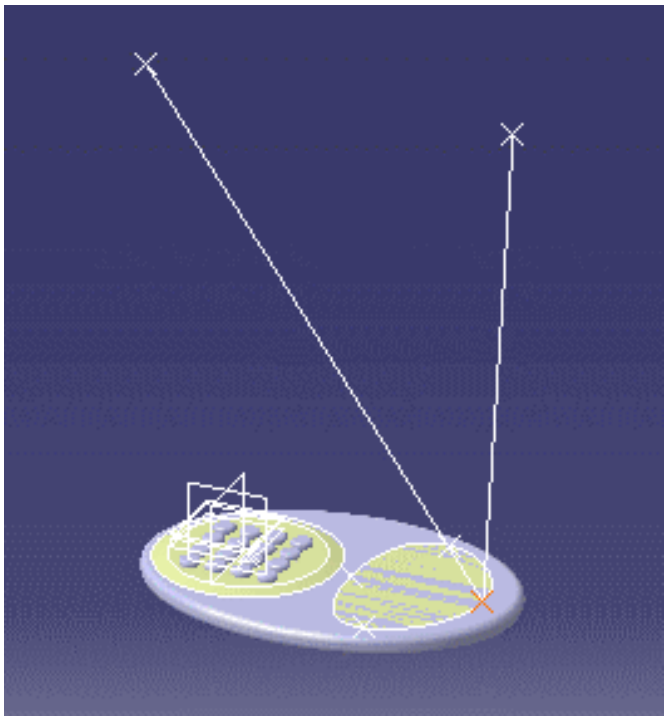
4. Replace the current script with the one below:

```
let X = Input: Feature "Select WindowP3";
context Input : Feature "Select OBody"

P1 isa GSMPoint
  {
   PointType = 0;
   TypeObject isa GSMPointCoord
    {
     X = 50mm;
     Y = 0mm;
     Z = 150mm;
    }
   }
P2 isa GSMPoint
   {
    PointType = 0;
    TypeObject isa GSMPointCoord
     {
      X = 50mm;
      Y = 100mm;
      Z = 150mm;
     }
    }

Line1 isa GSMLine
    {
     PointType = 0 ;
     TypeObject isa GSMLinePtPt
      {
       FirstPoint = object : Input X;
       SecondPoint = object: ../../P1;
      }
     }

Line2 isa GSMLine
     {
      LineType= 0 ;
      TypeObject isa GSMLinePtPt
       {
       FirstPoint = object : Input X;
       SecondPoint = object: ../../P2;
         }
      }
```

5. Click **Generate**. The dialog box below is displayed:



6. Click **Select** then select the WindowP3 feature. Click **Apply**. The second input line is highlighted. Click **Select** then select the OBody feature. Click **Apply** then click **OK**. Two lines are created. WindowP3 is the point where they intersect.



Tips and Tricks

About 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.

Example

The "mechanical" part of the PktPocketCalculator.CATGScript sample is to be generated in two shots:

1. import the necessary sketches and generate the pads:

```
import Input : FilePath "Select the PktInitialSketch.CATPart sample" ;

calc isa CATPart
{
part1 isa Part
{

partbody isa BodyFeature
{
S1 isa Sketch.1 {}
S2 isa Sketch.2 {}

pad1 isa Pad("S2")
{
FirstLength = 0.5mm;
SecondLength=2.0mm;
}

fillet1 isa ConstantEdgeFillet ("Face:(Brp:(pad1;0:(Brp:(S2;1)));None:())")
{
Radius =1.0 mm;
}

pattern1 isa Pattern[4,4] of fillet1
{
Step1 =7.0 mm;
Step2 = 7.0 mm;
}
/*
pad0 isa Pad("S1")
{
FirstLength = 5.0mm;
SecondLength=0mm;
}
*/
/*
fillet2 isa Fillet (select the cylindrical face of pad0 )
{
Radius =2.0 mm;
}
*/
}
}
}
```

2. Remove the comments before the following lines and click **Generate**:

```
pad0 isa Pad("S1")
```

```
{
FirstLength = 5.0mm;
SecondLength=0mm;
}
```

3. Capture the fillet1 and fillet2 definitions, remove the comments and re-execute the script:

```
import Input : FilePath "Select the PktInitialSketch.CATPart sample" ;

calc isa CATPart
{
part1 isa Part
{

partbody isa BodyFeature
{
S1 isa Sketch.1 {}
S2 isa Sketch.2 {}

pad1 isa Pad("S2")
{
FirstLength = 0.5mm;
SecondLength=2.0mm;
}

fillet1 isa ConstantEdgeFillet (capture the face using the Get surface command)
{
Radius =1.0 mm;
}

pattern1 isa Pattern[4,4] of fillet1
{
Step1 =7.0 mm;
Step2 = 7.0 mm;
}

pad0 isa Pad("S1")
{
FirstLength = 5.0mm;
SecondLength=0mm;
}

/*
fillet2 isa ConstantEdgeFillet (capture the face using the Get surface command )
{
Radius =2.0 mm;
}
*/
}
}
}
```

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.

The script below displays the message: "property does not exist: CstEdgeRibbon.2 on: EdgeFillet.1" at generation

```
Boite isa CATPart
{
boite isa Part
{
partbody isa BodyFeature
{
pad1 isa Pad
{
FirstLength = 0.5mm;
SecondLength=2.0mm;
}

fillet1 isa ConstantEdgeFillet ( "Face:(Brp:(pad1;2);None:())" )
{
CstEdgeRibbon.2\ Radius =1.0 mm;
}
}
}
}
```

Message "*feature* could not be updated"

This message is displayed whenever the system does not find the required data to build the specified feature.

In the PktPocketCalculator.CATGScript sample, replacing the WindowCurve and Fill2 definitions with the script below displays the message "Fill2 could not be updated".

```
WindowCurve isa GSMCurve
 {
  Elements[1] = object : ..\WindowP1;
  Elements[2] = object : ..\WindowP2;
 // Elements[3] = object : ..\WindowP3;
  Elements[4] = object : ..\WindowP4;
  Elements[5] = object : ..\WindowP1;
 }

Fill2 isa GSMFill
 {
  Boundary = object : ..\WindowCurve;
 }
```

Fill2 cannot be created because one element of the list required to defined the WindowCurve feature has been removed. WindowCurve is created but it is not a closed curve and it cannot be filled.

Features that "could not be generated" are displayed as invalid features in the specification tree. Editing an invalid feature can help you determine which data is missing. Another way to investigate consists in rebuilding interactively the feature that could not be built by the script.

Specifying a File Path (3 methods)

Method 1: Use the Insert File Path command from the contextual menu.

To do this, position the cursor where the file path is to be specified, then right-click and select the Insert File Path command from the contextual menu. In the dialog box which is displayed, select the appropriate path, then click Open. This insert the full path between quotation marks into your script.

Method 2: Define your linked document strategy.

Use the **Link Document Localization** command of the *CATIA* **Tools->Options...** menu to define your linked document strategy. Choosing an appropriate strategy allows you to specify only the short path of a document.
Example:
If the E:\www\samples folder is specified in the 'Search Order' of the 'Other Folders' Configuration, you can write:

```
import "PktInitialSketch.CATPart" ;
```

instead of

```
import "E:\www\samples\PktInitialSketch.CATPart" ;
```

See to the *CATIA* Infrastructure User's Guide for how to use the Link Document Localization command.

Method 3: Use the Input : FilePath keyword.

Input : FilePath *comment* ;
displays a dialog box which prompts you to select a file path. The file path being selected, you must click Apply to enter the next input data or make the OK button active.

The Rectangular Pattern is not Generated

Sorry! The present version of the Generative Script only allows you to generate rectangular pattern in the xy plane.

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.

Here is an example:

```
RuleBody =`if P->Name() == "Wing" Message("This is a wing")`;
```

Use Cases

Assembly Template	The Tow Hook
Scripting Template	The Ladder The Pocket Calculator

The Tow Hook

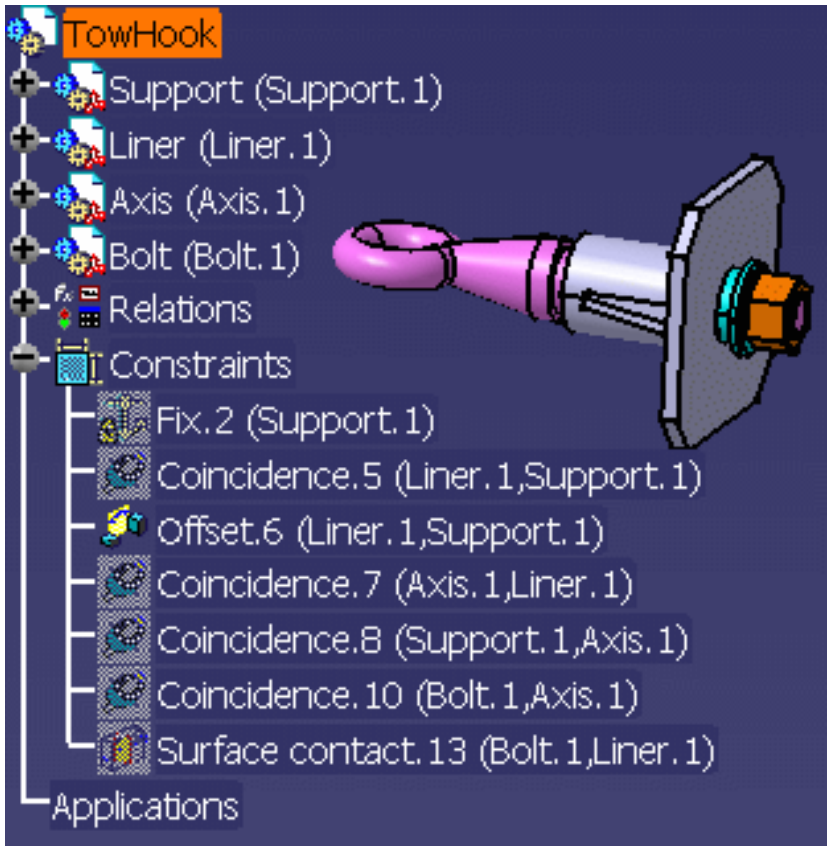
The scenario developed below is intended to show how to create and instantiate an assembly template into a .CATProduct file.

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

PktTowHook.CATProduct	M39.CATPart
	Liner_Step3.CATPart
PktTowHook_Result.CATProduct	Axis_Step3.CATPart
	Support.CATPart
PktDestinationProduct.CATProduct	Input_Axis.CATPart
	Part32.CATPart

Creating the Assembly Template

1. Open the PktTowHook.CATProduct file. The following image displays.

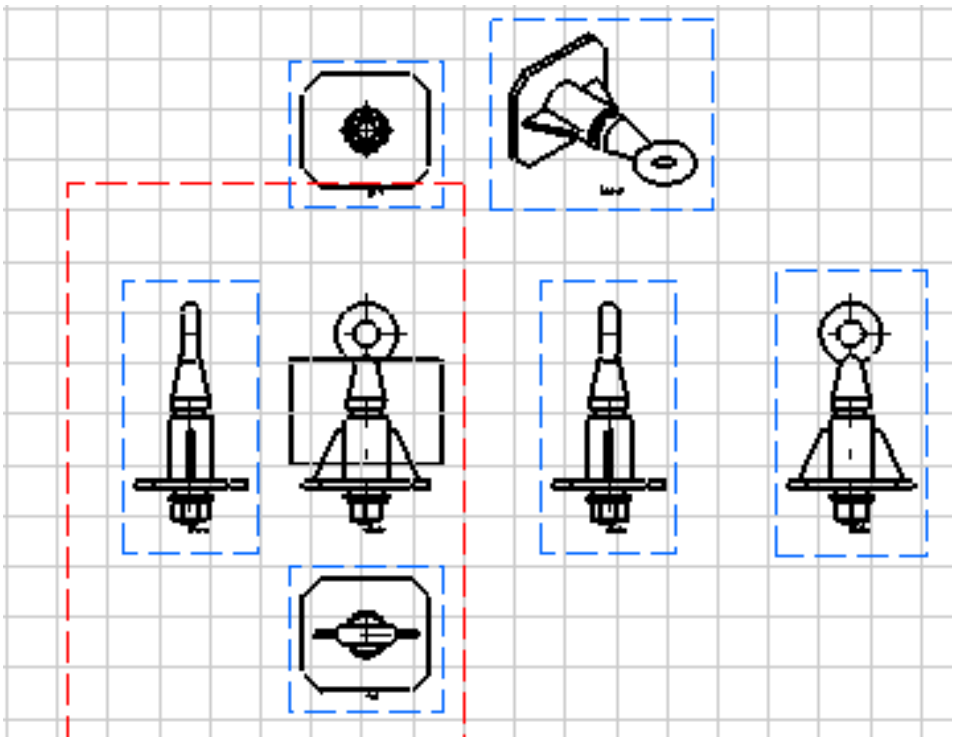


2. From the **Start->Mechanical Design** menu, access the **Drafting** workbench. The **New**

Drawing Creation Window displays.

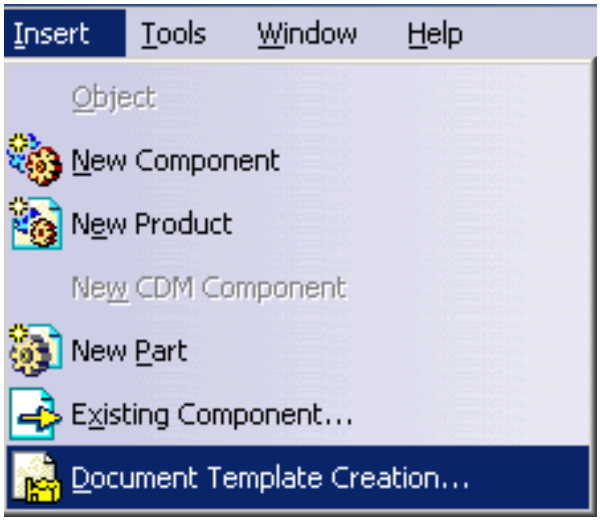
3. Select the **All views** configuration and click **OK**.

4. The drawing corresponding to the pad is generated.



4. Save your drawing and close the file. Click here to see the generated drawing.

5. Go back to the PktTowHook.CATProduct file to create an assembly template.
From the **Insert** menu, select the **Document Template Creation ...** command. The **Document Template Definition** window displays.



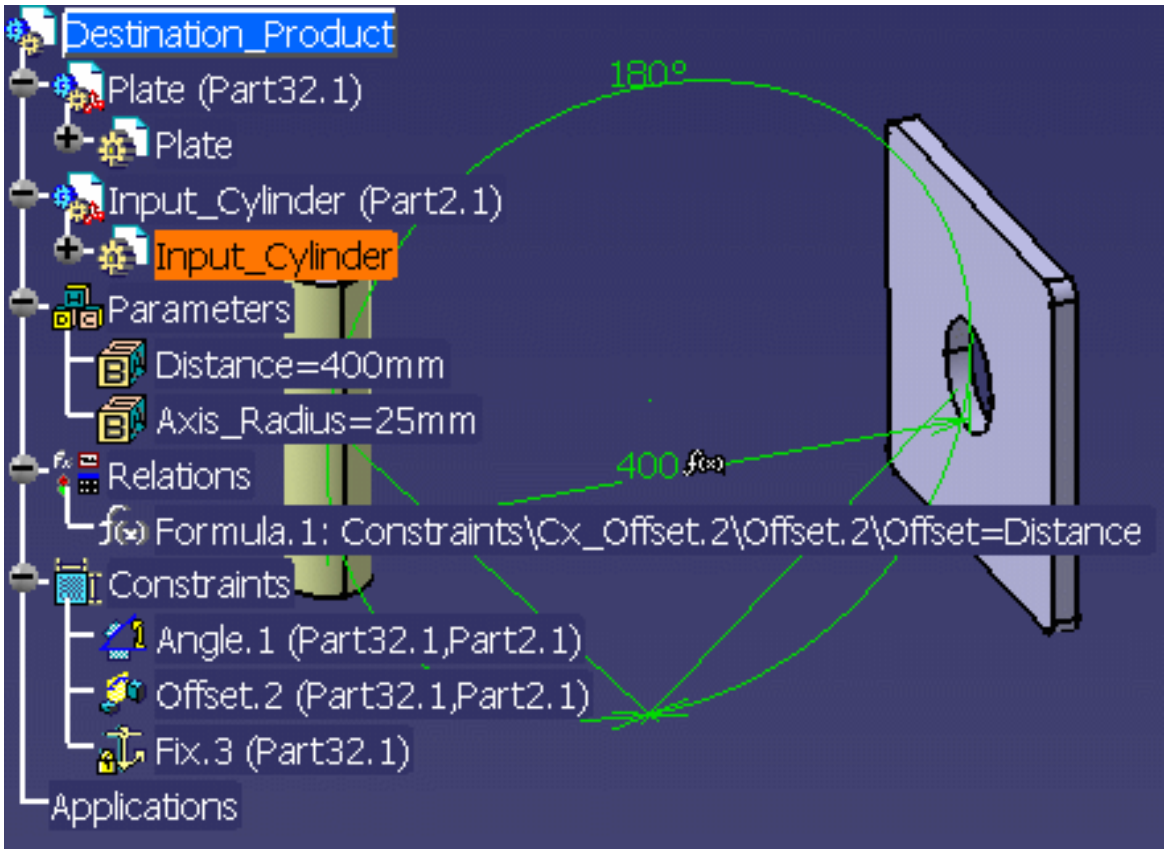
6. In the **Document Template Definition** window, define the document template. To do so, proceed as follows:

- In the **Documents** tab, click the **Add...** button and select the drawing that you have just selected or use the PktHookDrawing.CATDrawing file.
- Click the **Inputs** tab. In the geometry, expand the Support node and select the following items located below the Isolated External References node:
    - Surface.1
    - Curve.1
    - Surface.2
- Select Surface.1 and assign it a new name: PlanarFace
- Select Curve.1 and assign it a new name: Center_CircularEdge
- Select Surface.2 and assign it a new name: Axis_CylindricalFace
- Click the **Published Parameters** tab and click the **Edit List...** button.
- In the **Select parameters to insert** window, select the Support\Tube_Thickness parameter using the arrow button.
- Click **OK** to validate. The Document template displays below the KnowledgeTemplates node.

7. Save your file and close it. Click here to open the result .CATProduct file.

Instantiating the Assembly Template

8. Open the PktDestinationProduct.CATProduct file. The following image displays.



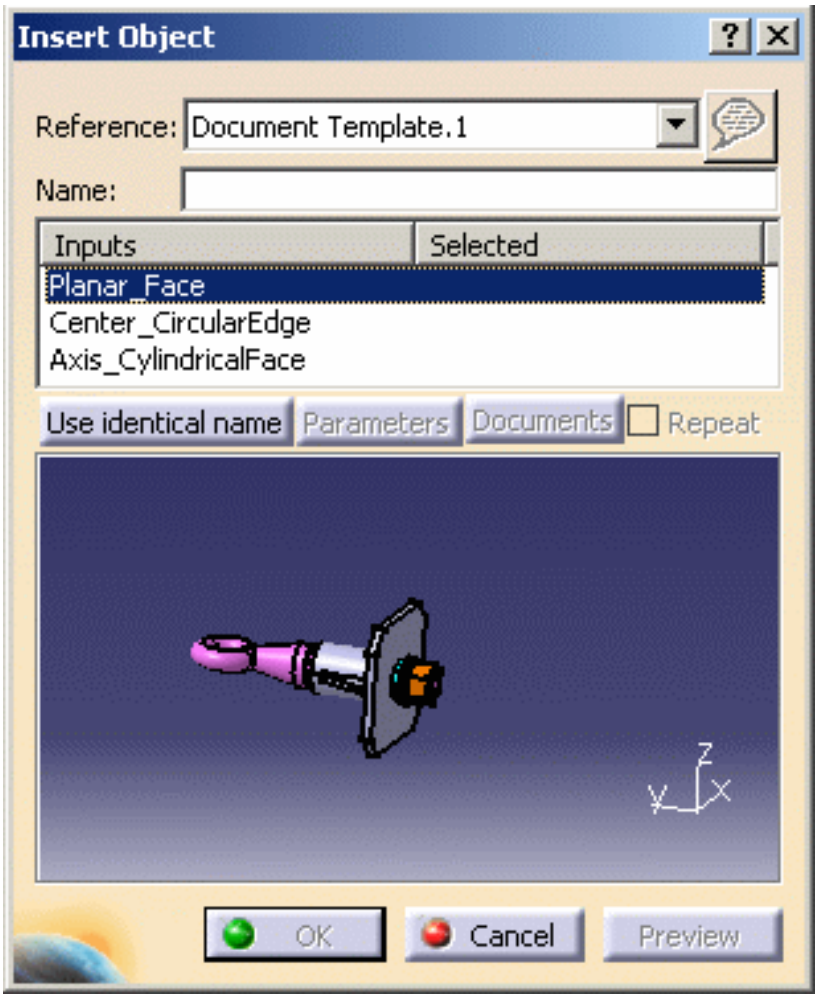
9. From the **Start->Knowledgeware** menu, access the Product Knowledge Template workbench.

10. Click the **Instantiate from Document** icon ( ) and select the PktTowHook_result.CATProduct file in the **File Selection** window. The Insert Object dialog box displays.

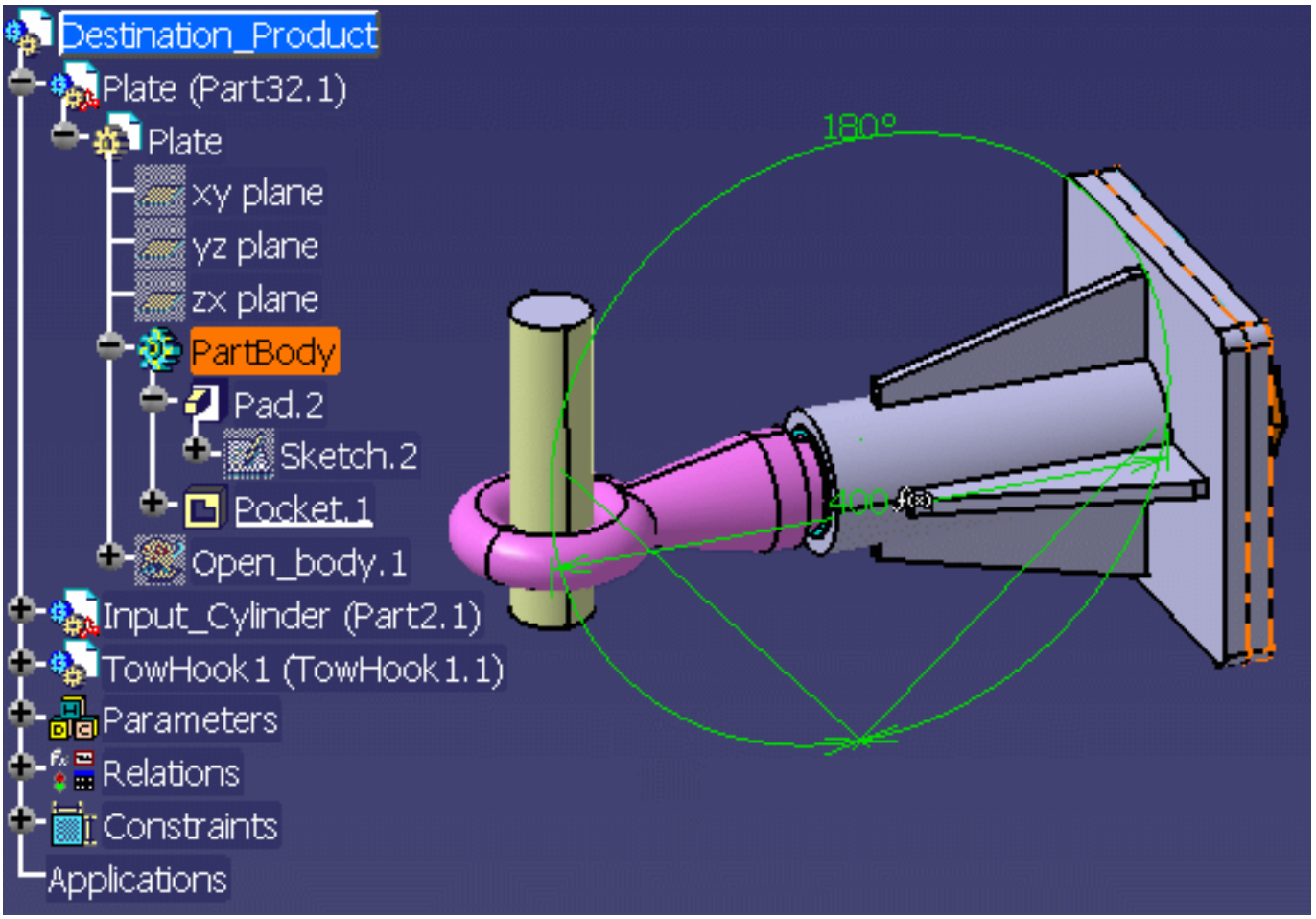
11. Select the visible face of the pad. Face displays in the Insert Object dialog box.

12. Select the pocket in the Geometry.

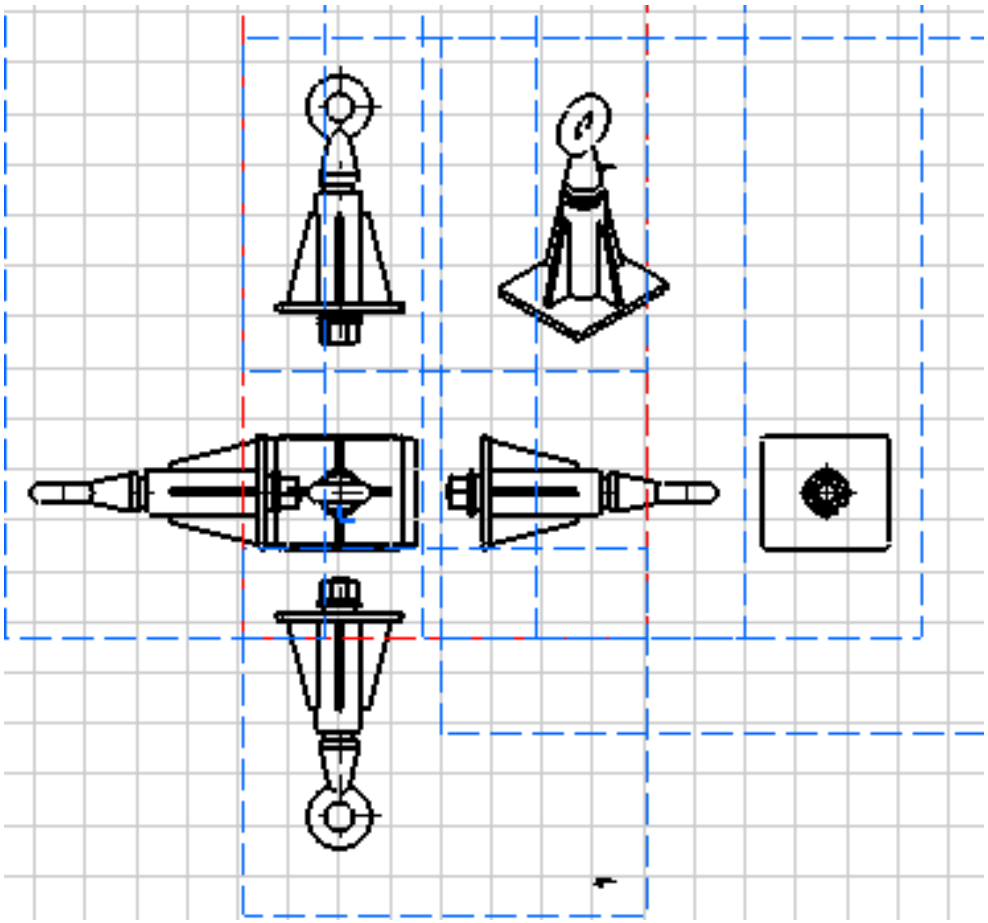
13. Expand the Input_Cylinder node and select the Extract.1.



14. Click **OK** to validate. The assembly template is instantiated (Click here to open the result file)...



and the associated drawing is updated accordingly (click here to open the generated drawing).



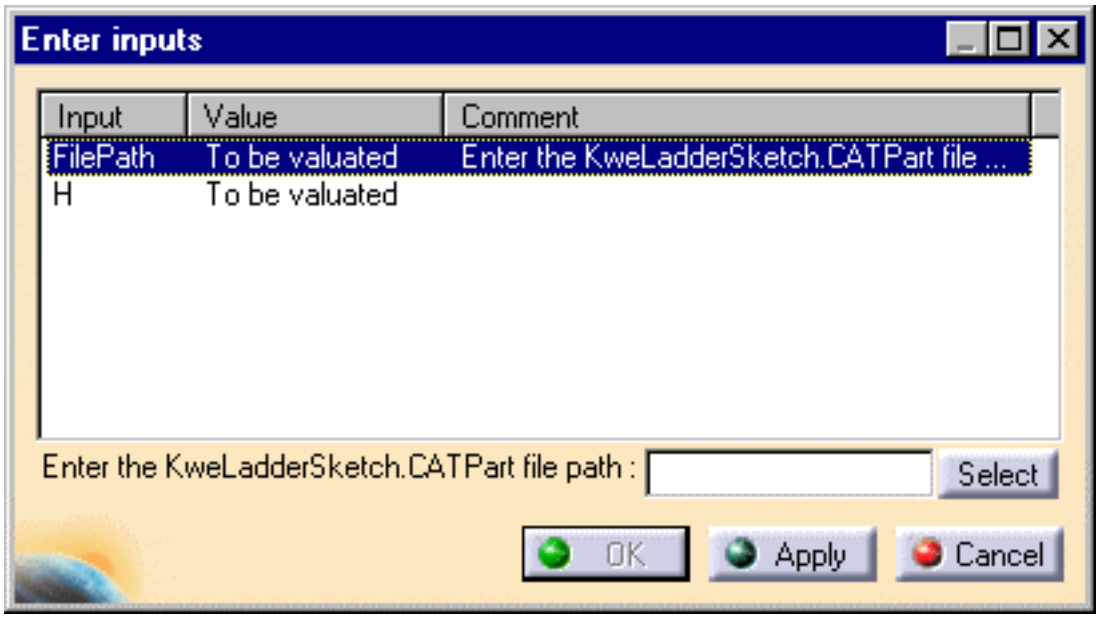
The Ladder

The scenario below illustrates the following generative script capabilities:

- Multiple value parameters
- Part design patterns
- Knowledge advisor rules
- Formulas using the ? operator.

To carry out this scenario, you need the PktLadderSketch.CATPart sample which provides you with the necessary sketches.

1. Select the **Knowledgeware->Product Knowledge Template** command from the Start menu.
2. Click the  icon. The Knowledgeware Script Editor is displayed.
3. Use the **File->Open** command to open the PktLadder.CATGScript document, then click **Generate**. The **Enter Inputs** dialog box below is displayed.



4. Click **Select**, then select the PktLadderSketch.CATPart sample in the file selection box. Click **Apply**. The H input is now highlighted. The H field displays a three value list.
5. Select one of the values, 3500mm for example. Click **Apply**, then click **OK**. The document is generated.

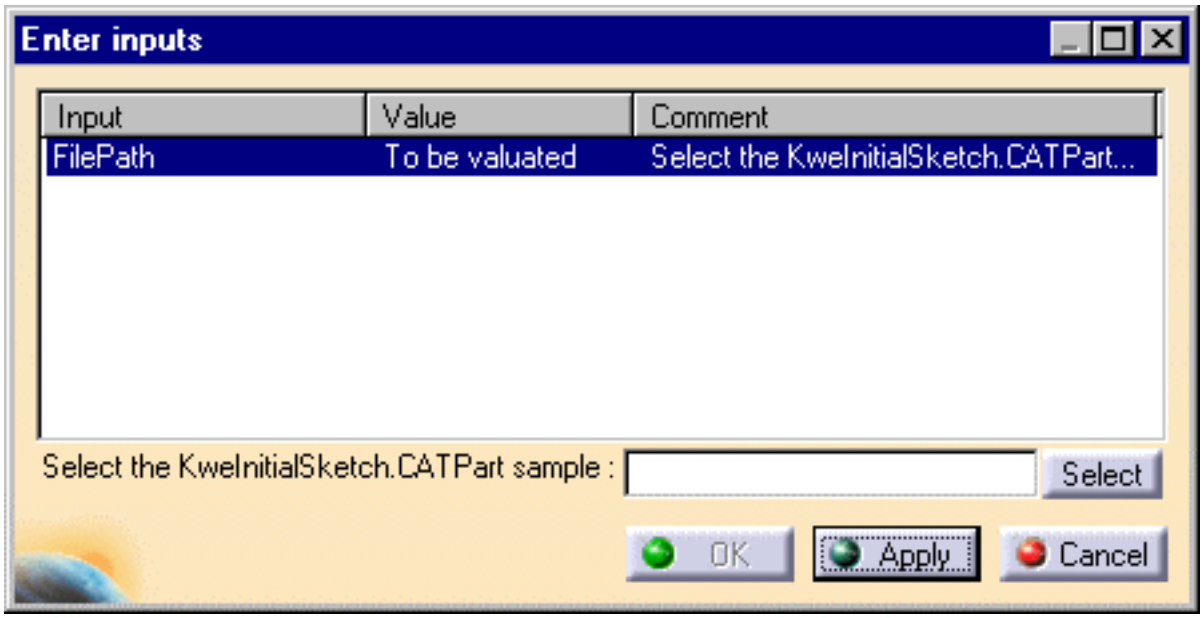
The Pocket Calculator

The scenario developed below is intended to show how to combine part design and shape design features to generate a simple pocket calculator. It comprises the steps and instructions required by the user for each step.

To carry out the scenario below, you need to have on hand the basic sketches from which you create your mechanical features. These basic sketches are available in the PktInitialSketch.CATPart sample. The shape design features are created from scratch.

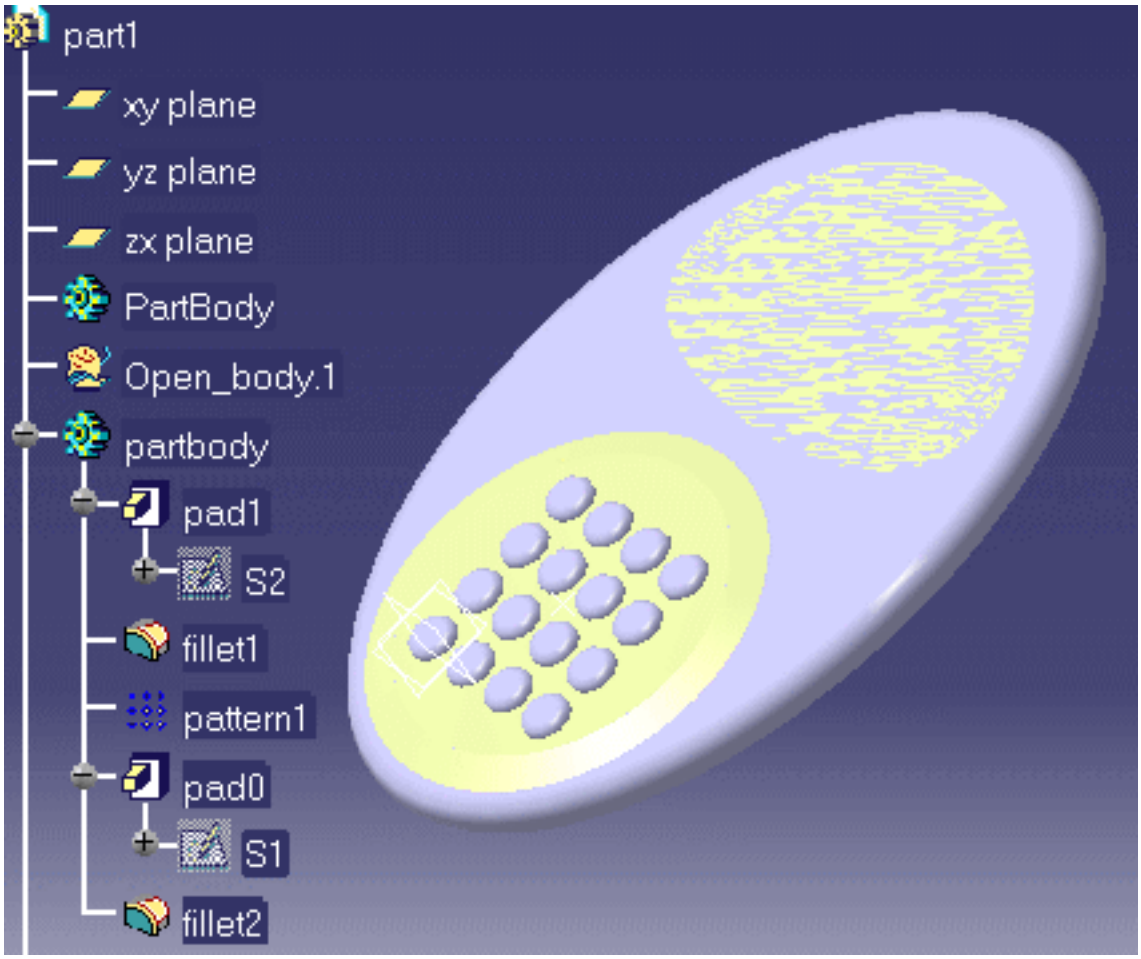
To carry out the scenario below, we use the PktInitialSketch.CATPart sample (~**46KB**) along with the PktPocketCalculator.CATGScript script (~**8KB**). Check the size of the resulting document. It should be around **250KB**.

1. Select the **Knowledgeware->Product Knowledge Template** command from the Start menu.
2. Click the  icon. The Knowledgeware Script Editor is displayed.
3. Use the **File->Open** command to open the PktPocketCalculator.CATGScript document, then click **Generate**. The dialog box below is displayed:



4. Click **Select**. The **Insert File Path** dialog box is displayed. Select the PktInitialSketch.CATPart sample, then click Open. You are back to the **Enter Inputs** dialog box.
5. Click **Apply**, then click **OK**. The document generation starts.

This is what you get onscreen, once the generation process is over.



Back to the script:

- The **import** statement below displays an input box whereby you can specify a sketch file. The pads created in the script all rely on these sketches.

```
import Input:FilePath "Select the PktInitialSketch.CATPart sample";
```

- The script is divided into two sections:

a) The part design features are enclosed by the BodyFeature
b) The shape design features are enclosed by the OpenBodyFeature.

```
calc isa CATPart
{
part1 isa Part
{
partbody isa BodyFeature
{
// your mechanical features are described here
}

OBody isa OpenBodyFeature
{
// your shape design features are described here
}
}
}
```

The fillet and pattern type features cannot be generated in one shot. You must first of all create the pad0 and pad1 objects, then add to your script the statements necessary to generate fillets and patterns. See the Fillet Object.

Reference

The packages listed below are those displayed in the Script Browser.

Basic Wireframe Package	GSD Package
GSD Shared Package	Knowledge Expert
Mechanical Modeler	Part Design
Part Shared Package	Standard

Using the Object Browser

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 Product Knowledge Template.

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:

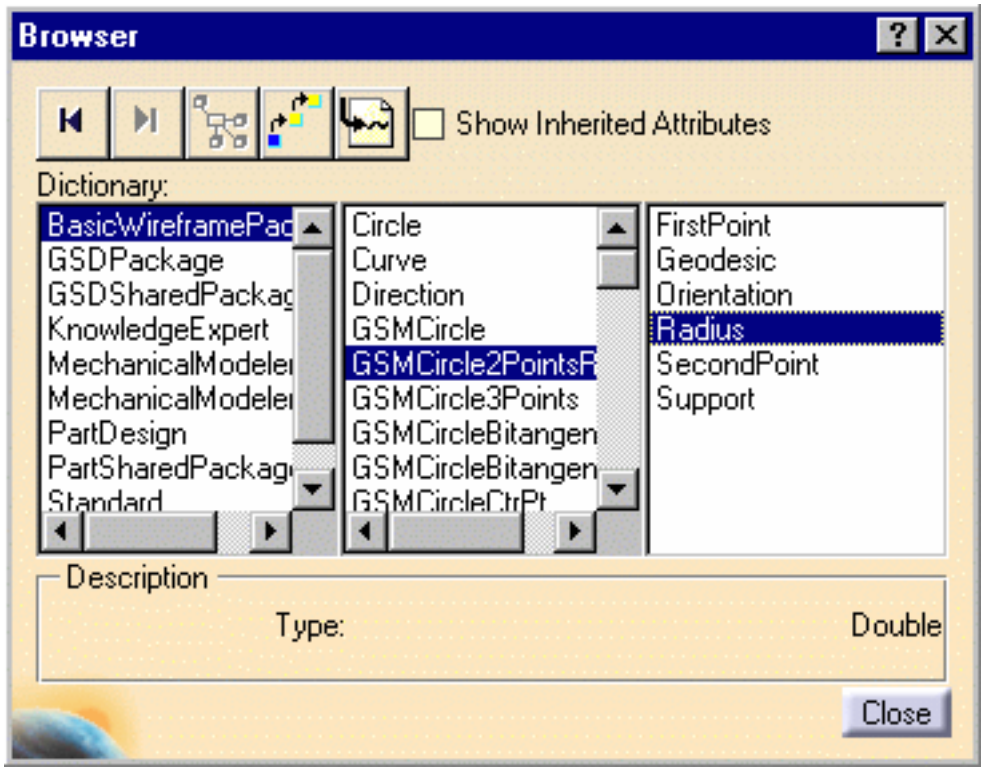
1. Select the **Tools->Options...** command to open the Options window, then select General->**Parameters and Measure**, and click the Language tab.
2. In the **Language** field of the **Knowledge** tab, check **Load extended language libraries** and select the libraries.

To access the Product Knowledge Template Object Browser, proceed as follows:

1. Access the Product Knowledge Template workbench by selecting the **Knowledgeware-> Product Knowledge Template** command from the **Start** menu.
2. Click the  icon. The Knowledgeware Script Editor is displayed.
3. Select the **Tools->Object Browser...** command from the Script Editor toolbar. The following window opens:

From this window, you can manipulate the list of objects supported by Generative Knowledge using their attributes.

- The left part of the browser displays the packages available: BasicWireframePackage, GSDPackage, GSDSharedPackage, KnowledgeExpert, MechanicalModeler, PartDesign, PartSharedPackage, Standard.



- 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).

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.

Basic Wireframe Package

GSMCircle
GSMLine
GSMPlane
GSMPoint

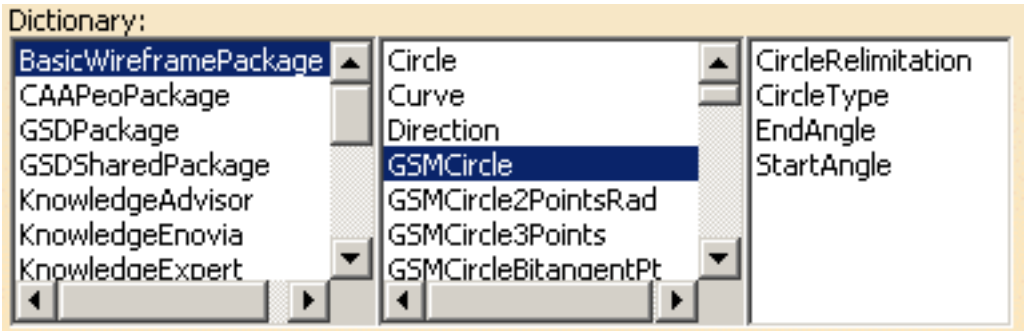
GSMCircle

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.

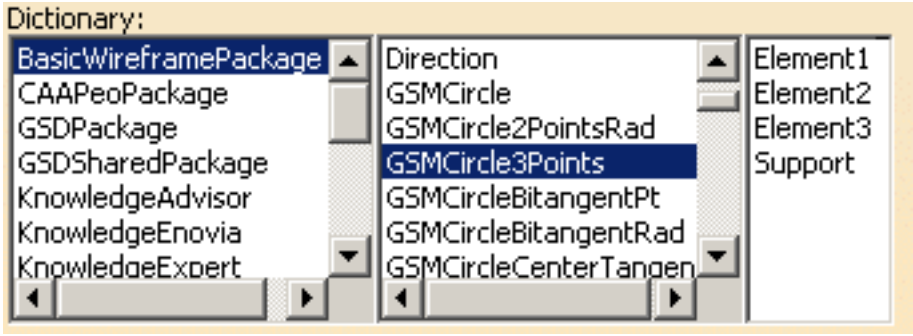
Plane Type in GSD	Plane Type in the Package	Corresponding digit
Three Points	GSMPCircle3Points	3
Center and Radius	GSMCircleCtrRad	0
Center and Point	GSMCircleCtrPt	1

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:

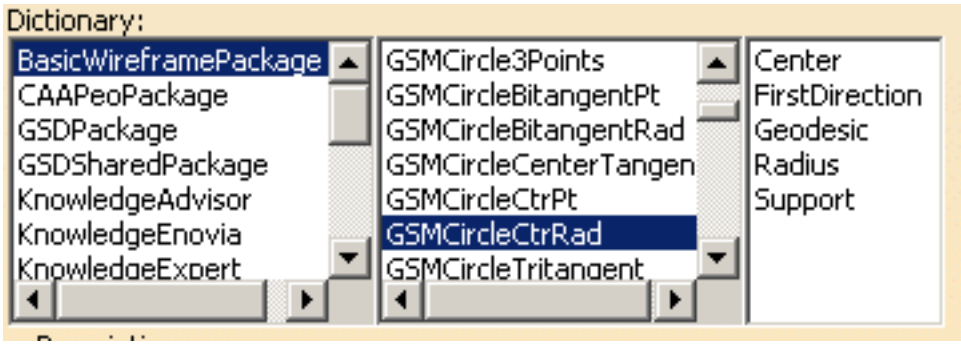
Combination

- 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:

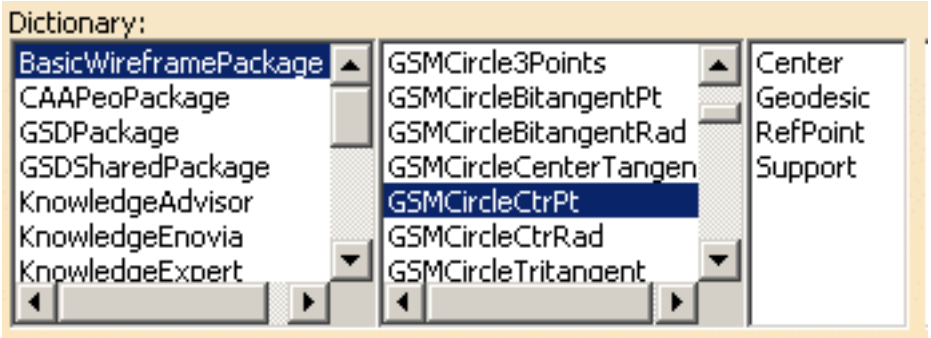
Combination

- 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;`

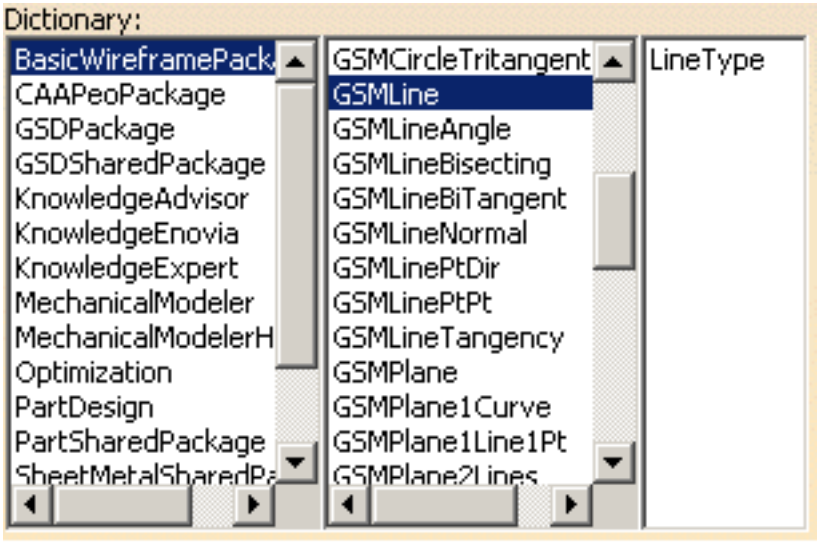
GSMLine

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.

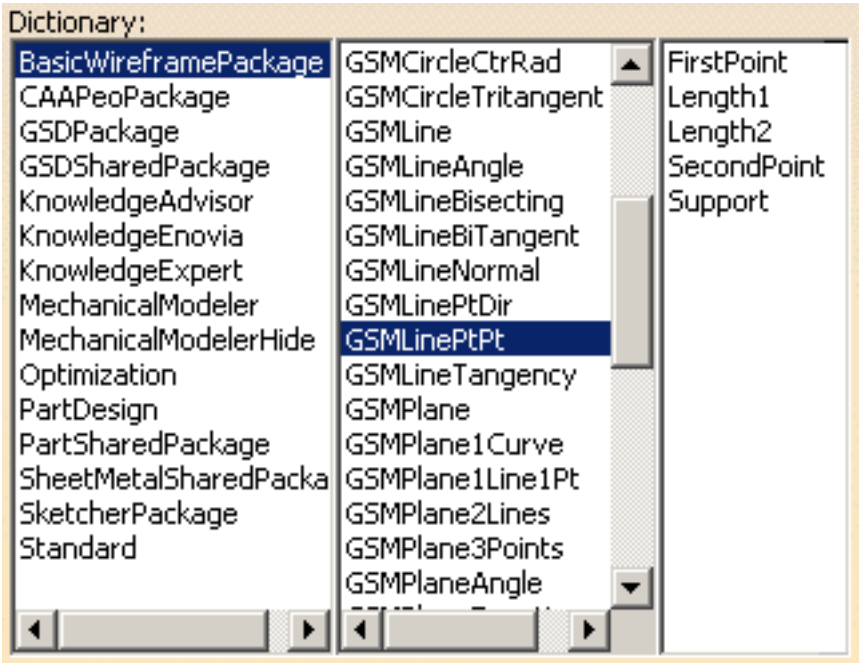
Line Type in GSD	Line Type in the Package	Corresponding digit
Point to Point	GSMLinePtPt	0
Point-Direction	GSMLinePtDir	1
Angle to Curve	GSMLineAngle	2
Tangent to Curve	GSMLineTangency	3
Normal to surface	GSMLineNormal	4
Intersection betw. 2 planes	GSMLineBiTangent	5

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 *FirstPoint* which is defined by the syntax below: FirstPoint = object: ..\..*theFirstPoint*;
• the *SecondPoint* which is defined by the syntax below: SecondPoint = object: ..\..*theSecondPoint*;	• the *SecondPoint* which is defined by the syntax below: SecondPoint = object: ..\..*theSecondPoint*;
• Length1 which is defined by the syntax below: Length1=200mm;	• the Support
• Length2 which is defined by the syntax below: Length2=150mm;	

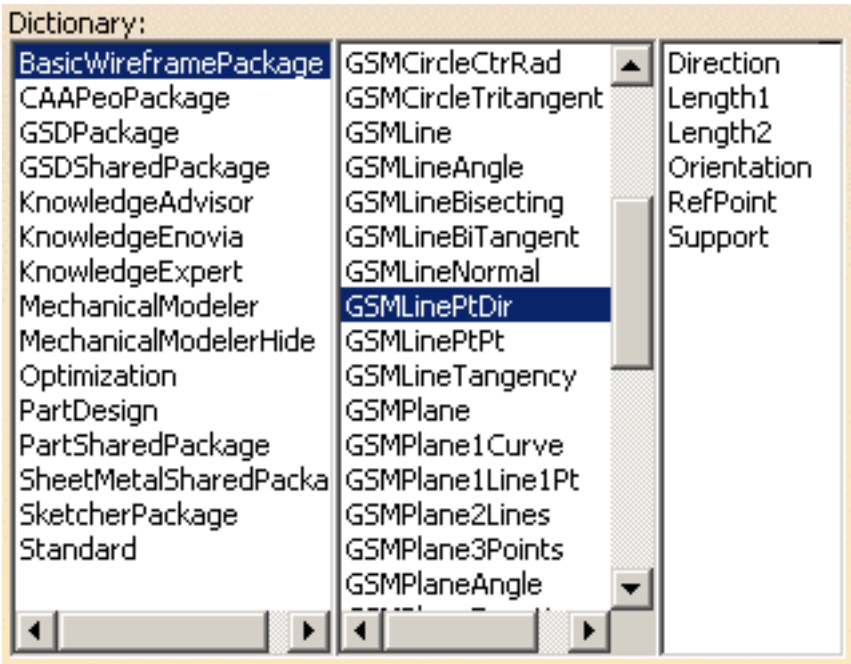
```
Line.1 isa GSMLine
{
    LineType = 0;
    TypeObject isa GSMLinePtPt
    {
     FirstPoint = object: Point.1;
     SecondPoint = object: Point.2;
     Length1 = 500mm; \\ optional
     Length2 = 435mm; \\ optional
    }
   }
```

```
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:

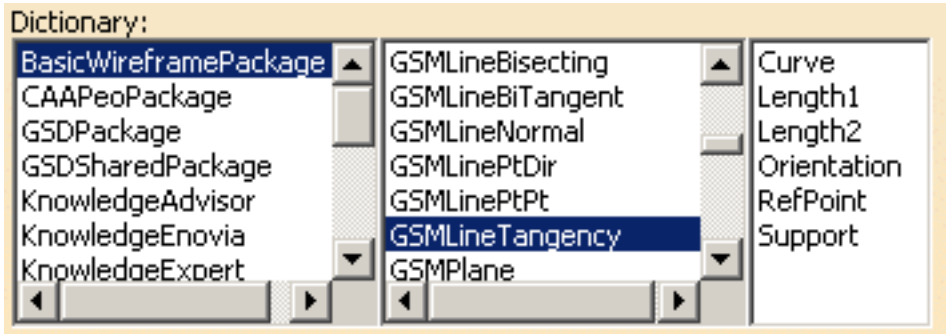
Combination

- 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:

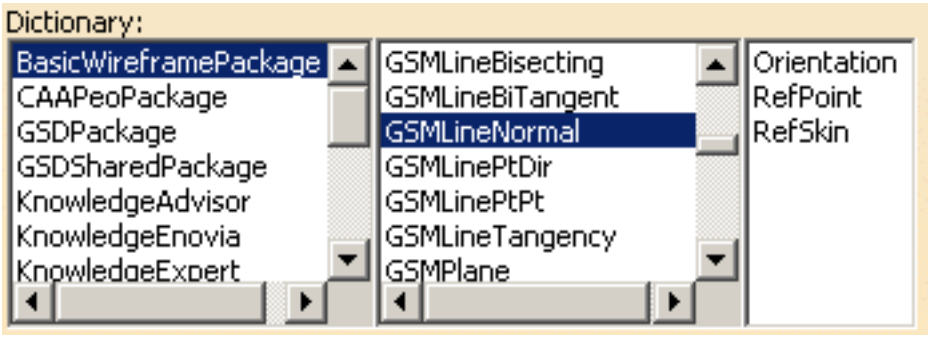
Combination

- 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

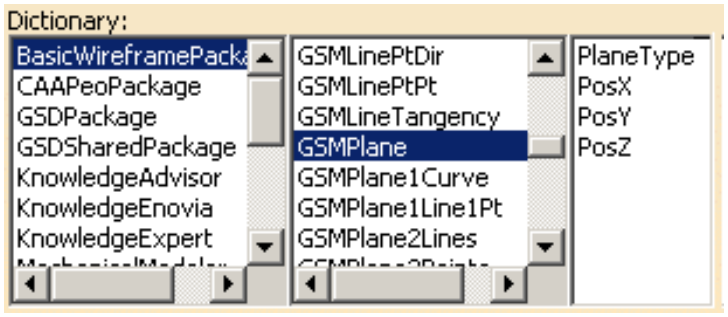
- *RefPoint* which is defined by the syntax below:
  RefPoint = object: ..\..\Point.2;
- Support which is defined by the syntax below:
  RefSkin = object: ..\..\Extrude.1;

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.

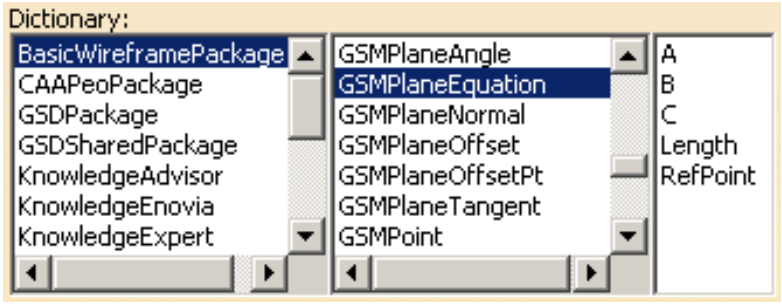
Plane Type in GSD	Plane Type in the Package	Corresponding digit
Equation	GSMPlaneEquation	0
Through 3 points	GSMPlane3Points	1
Through 2 lines	GSMPlane2Lines	2
Through a point and a line	GSMPlane1line1Pt	3
Normal to a curve	GSMPlane1Curve	4
Tangent to a surface	GSMPlaneTangent	5
Normal to a plane	GSMPlaneNormal	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 ;

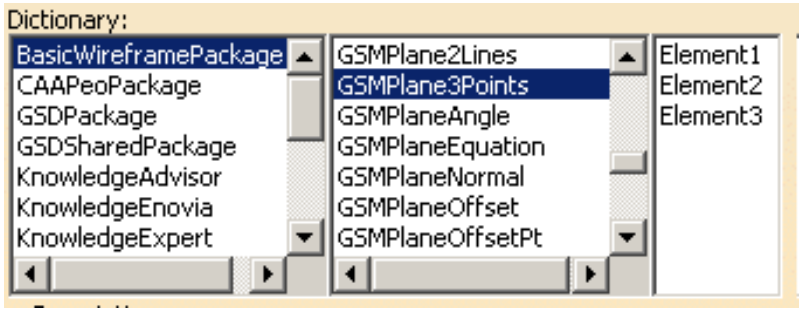
```
Plane0.1 isa GSMPlane
{
   PlaneType = 0;
   TypeObject isa GSMPlaneEquation
    {
      A = 15;
      B = -12;
      C = 31;
      Length = 24mm;
    }
}
```

```
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

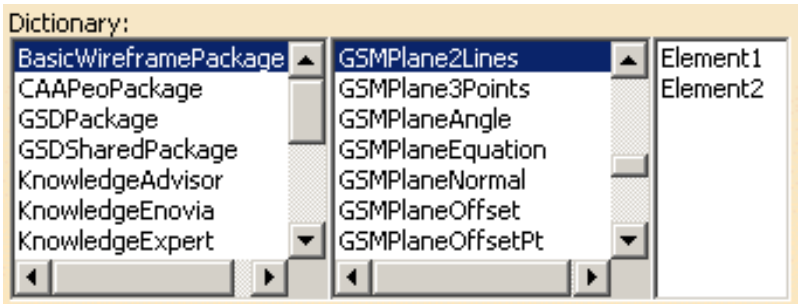
- Element1 which is defined by the syntax below:
  Element1 = object: ..\Point.1;
- Element2 which is defined by the syntax below:
  Element2 = object: ..\Point.2;
- Element3 which is defined by the syntax below:
  Element3 = object: ..\Point.3;

```
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

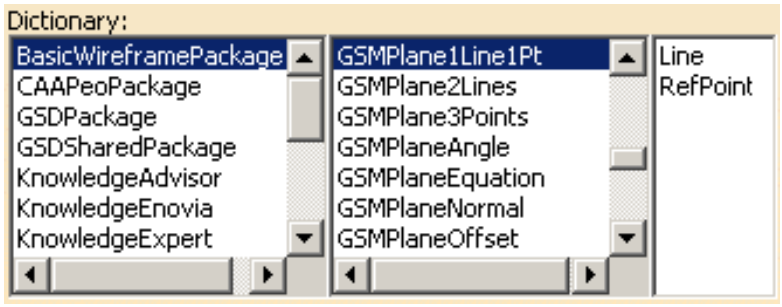
- Element1 which is defined by the syntax below:
  Element1 = object: ..\Line.1;
- Element2 which is defined by the syntax below:
  Element2 = object: ..\Line.2;

```
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

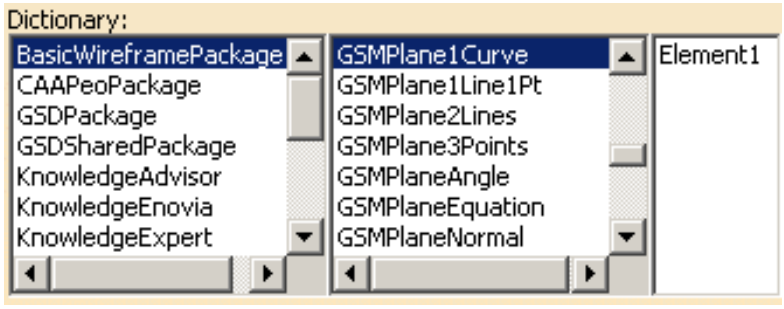
- Line which is defined by the syntax below:
  Line = object: ..\Line.1;
- RefPoint which is defined by the syntax below:
  RefPoint = object: ..\Point.2;

```
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:

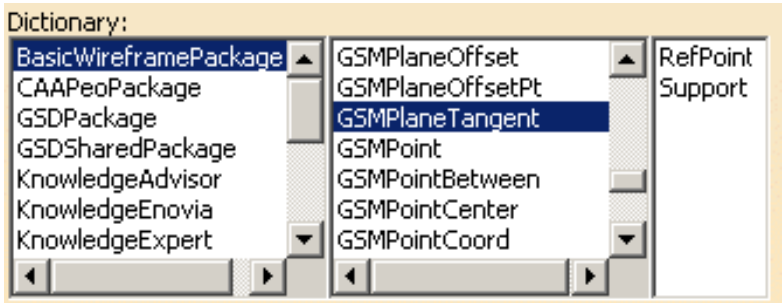
Combination

- 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

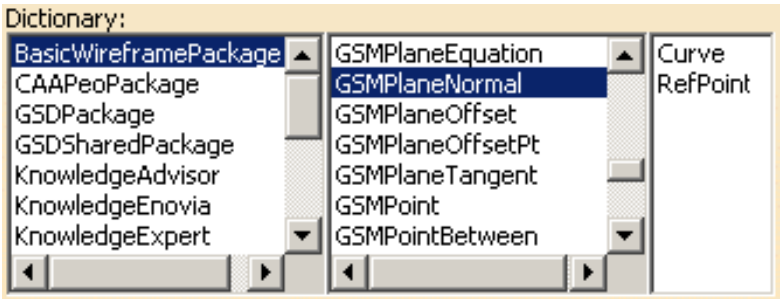
- Support which is defined by the syntax below:
  `Support = object: ..\Spline.1;`
- RefPoint which is defined by the syntax below:
  `RefPoint = object: ..\Point.4;`

```
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;

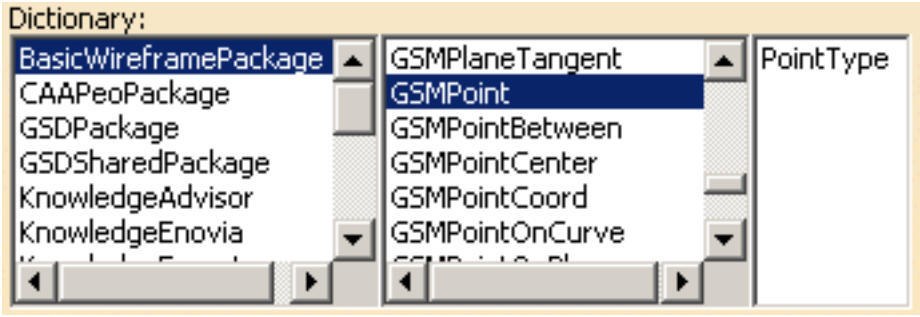
```
Plane6 isa GSMPlane
{
   PlaneType = 6;
   TypeObject isa GSMPlaneNormal
            {
                Curve = object: ..\Spline.2;
                RefPoint = object: ..\Point.13;
            }
}
```

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.

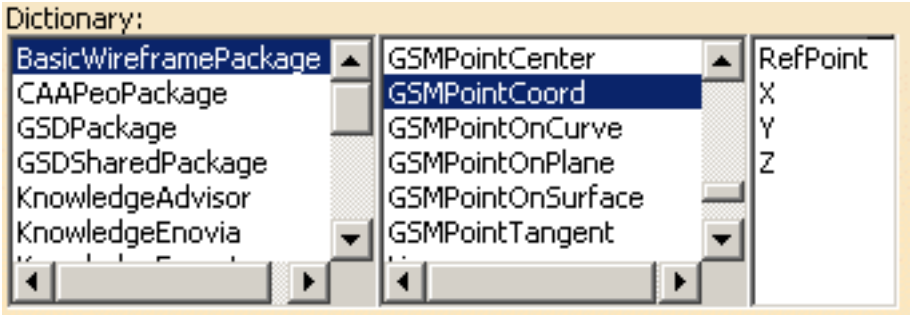
Plane Type in GSD	Plane Type in the Package	Corresponding digit
Coordinates	GSMPointCoord	0
On surface	GSMPointOnSurface	1
On curve	GSMPointOnCurve	2
On plane	GSMPointOnPlane	3
Circle center	GSMPointCenter	4

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

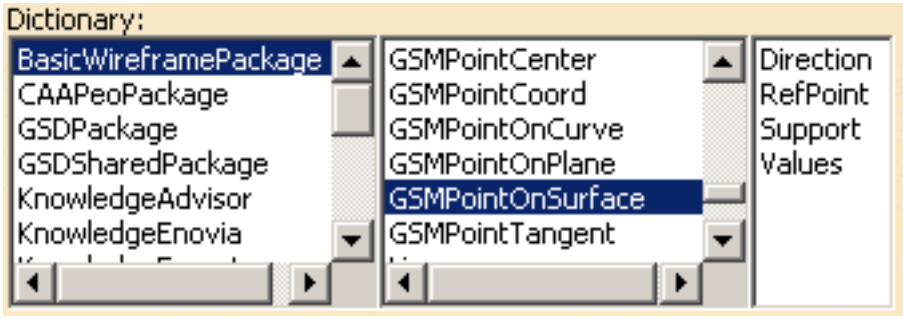
- RefPoint (Reference point, optional)
- X which is defined by the syntax below:
  X = 10mm;
- Y which is defined by the syntax below:
  Y = 10mm;
- Y which is defined by the syntax below:
  Z = 10mm;

```
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

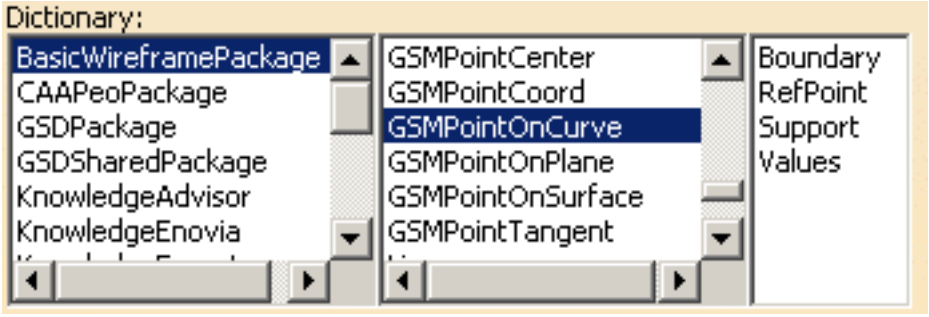
- Direction which is defined by the syntax below:
  Direction = object: ..\Line.1;
- Support which is defined by the syntax below:
  Support= object: ..\Extrude.1;
- Values which is defined by the syntax below:
  Values = 12mm;

```
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

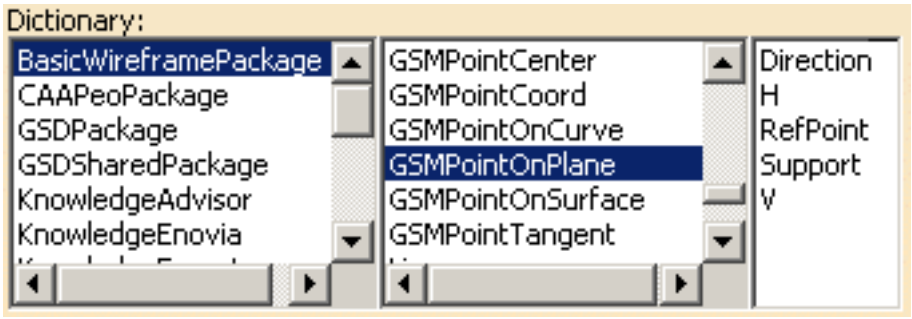
- Refpoint which is defined by the syntax below:
  RefPoint= object: ..\Point.1;
- Support which is defined by the syntax below:
  Support = object: ..\Line.1;
- Values which is defined by the syntax below:
  Values = 12mm;

```
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

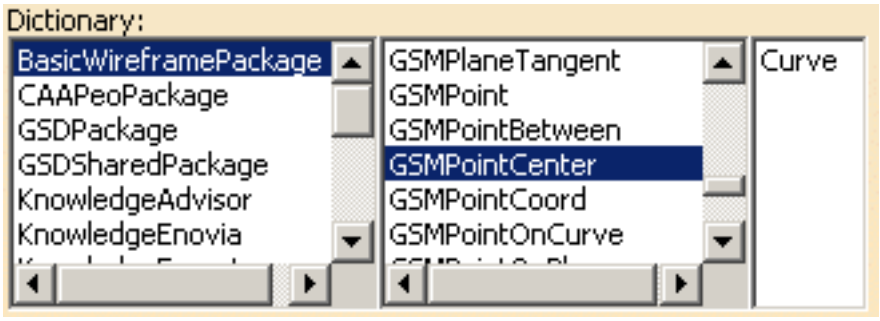
- Direction which is defined by the syntax below:
  Direction = object: ..\Line.1;
- H which is defined by the syntax below:
  H = 150mm;
- RefPoint which is defined by the syntax below:
  RefPoint= object: ..\Point.1;
- Support which is defined by the syntax below:
  Support = object: 'xy plane'
- V 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.

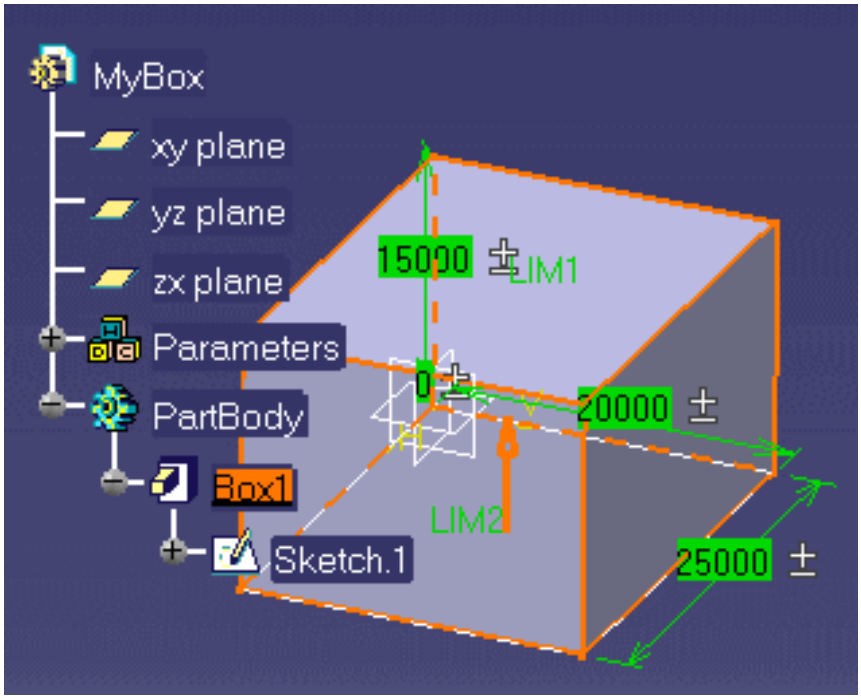
Attributes:

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 ;
       }
     }
   }
```

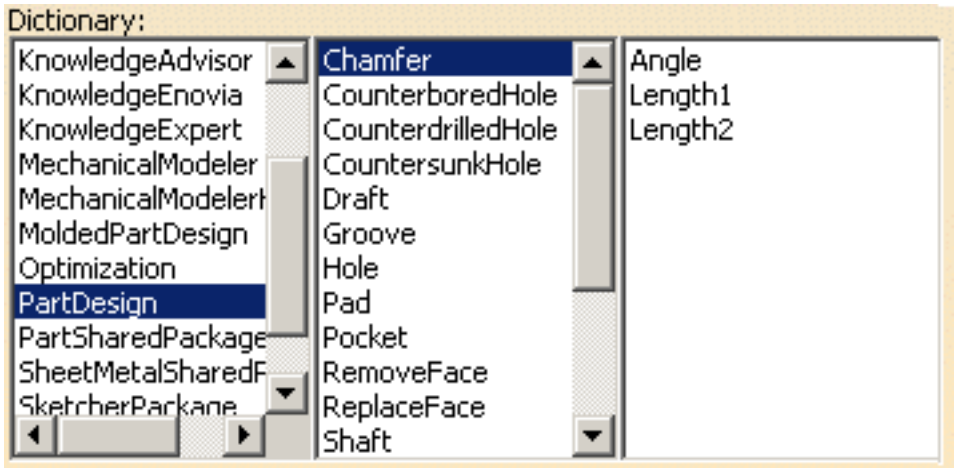
```
}
```



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.

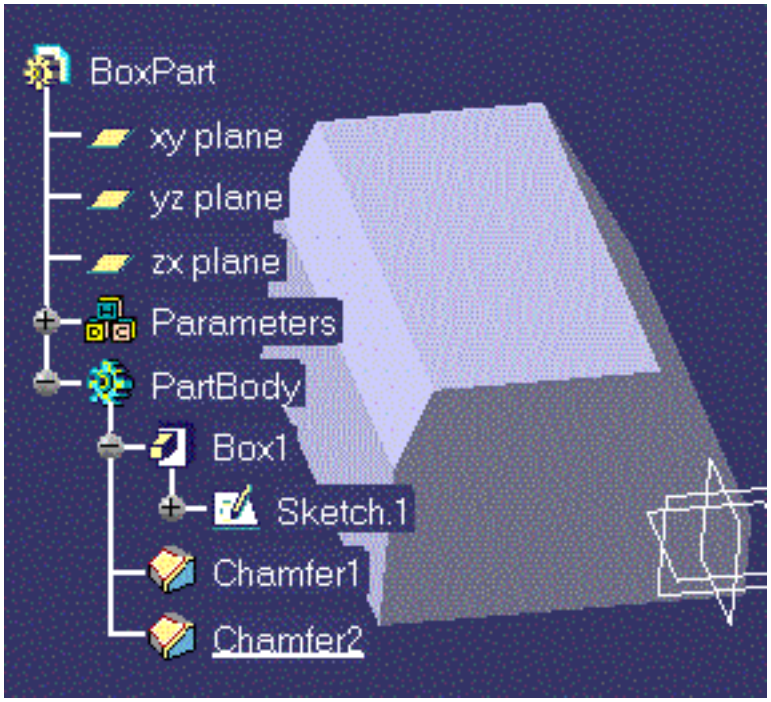
To specify a chamfer within your script, you must have a part open, then proceed as follows:

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:

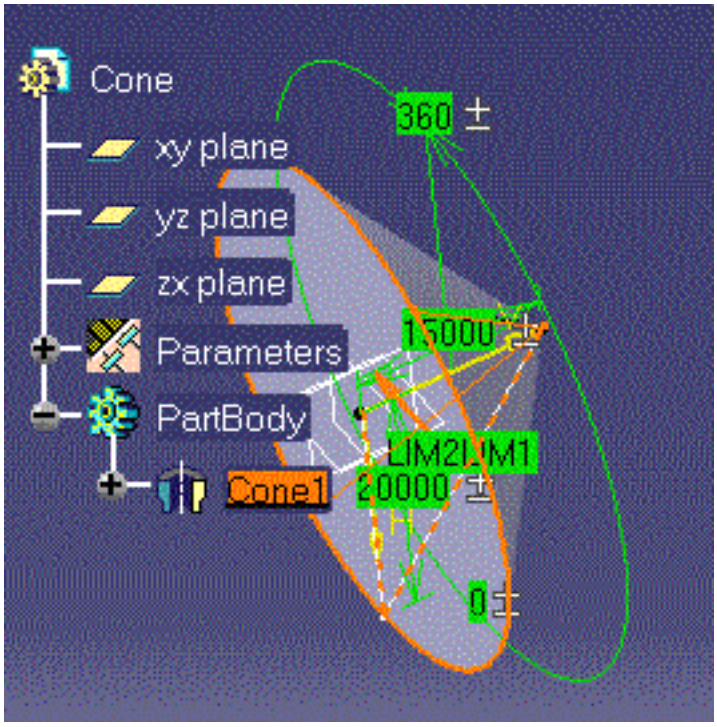
A cone is a shaft created by rotating a triangular sketch.

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 ;**.

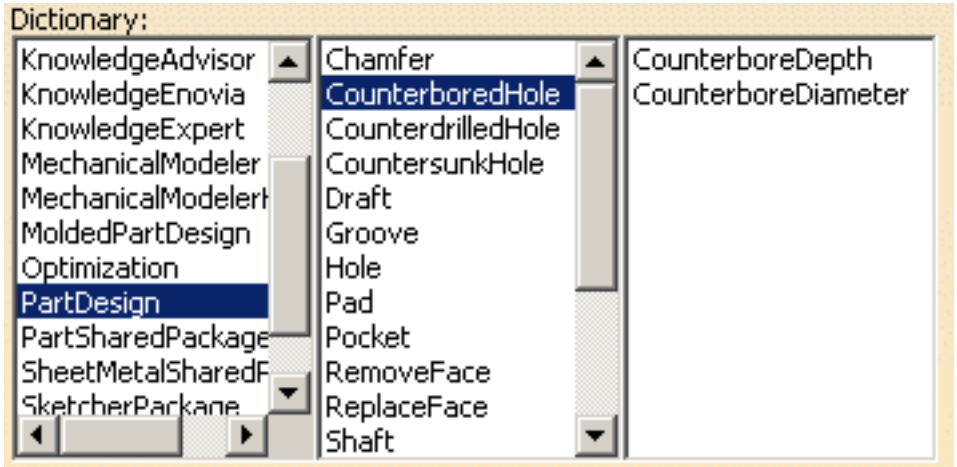
```
MyCone isa CATPart
  {
   ConePart isa Part
     {
      PartBody isa BodyFeature
        {
         // Create a cone
         Cone1 isa Cone
            {
             Radius = 20.0 mm ;
             Length = 15.0 mm ;
            }
        }
     }
 }
```



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:

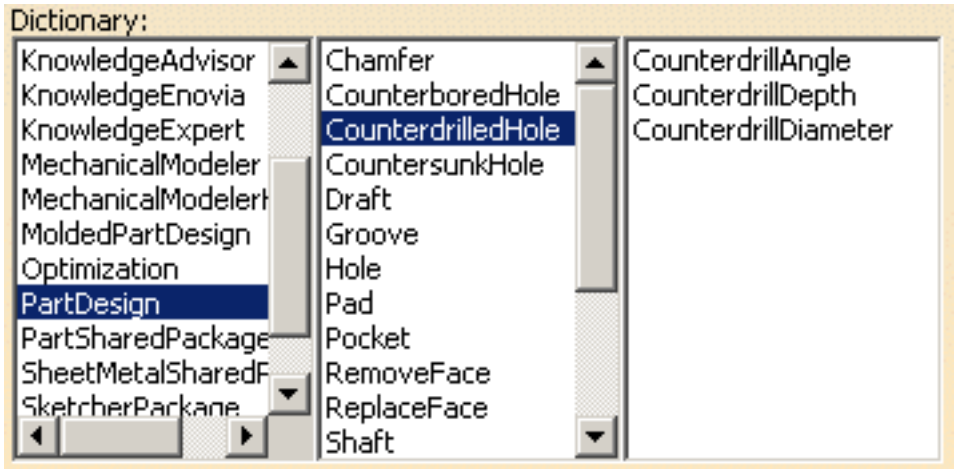
A counterbored hole is defined by the following attributes:

- 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:

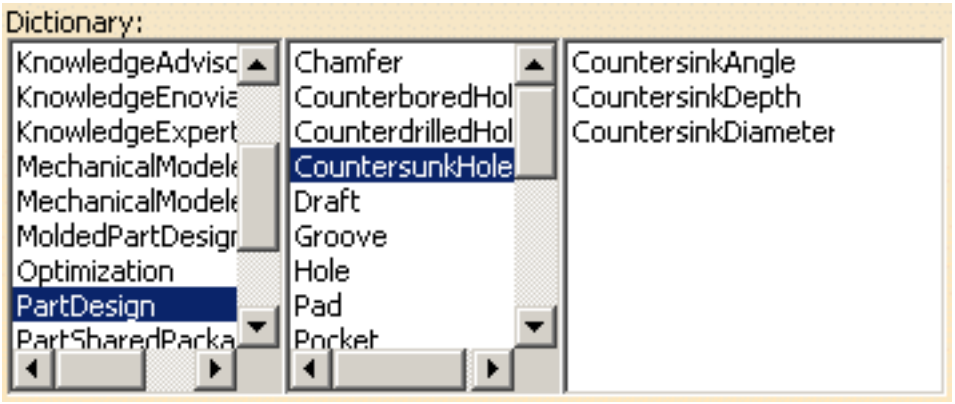
A counterdrilled hole is defined by the following attributes:

- 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:

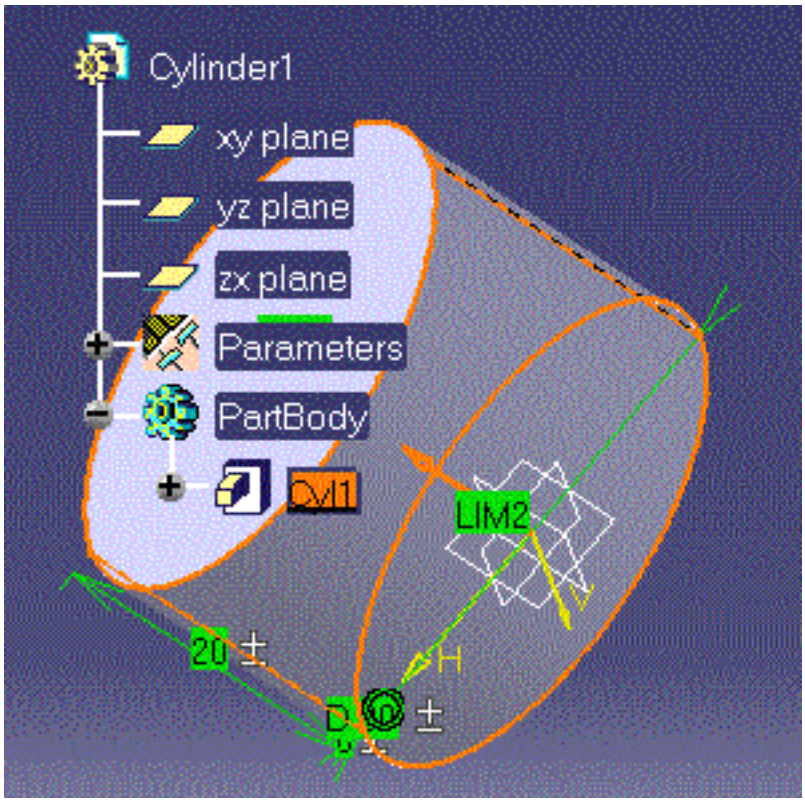
A cylinder is a pad created by extruding a circular sketch.

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**.

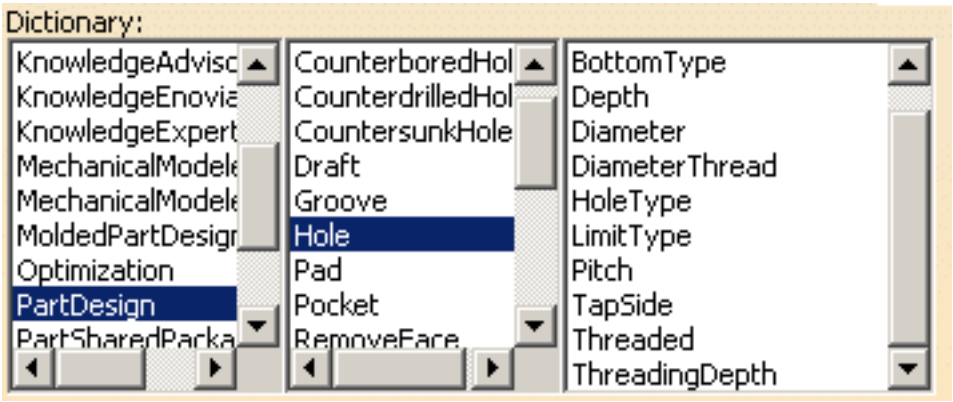
```
Cylinder1 isa CATPart
 {
  Part isa Part
   {
    PartBody isa BodyFeature
     {
      // Create a cylinder
      Cyl1 isa Cylinder
        {
         Radius=15.0 mm;
         EndLimit=20.0 mm;
        }
     }
   }
 }
```



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

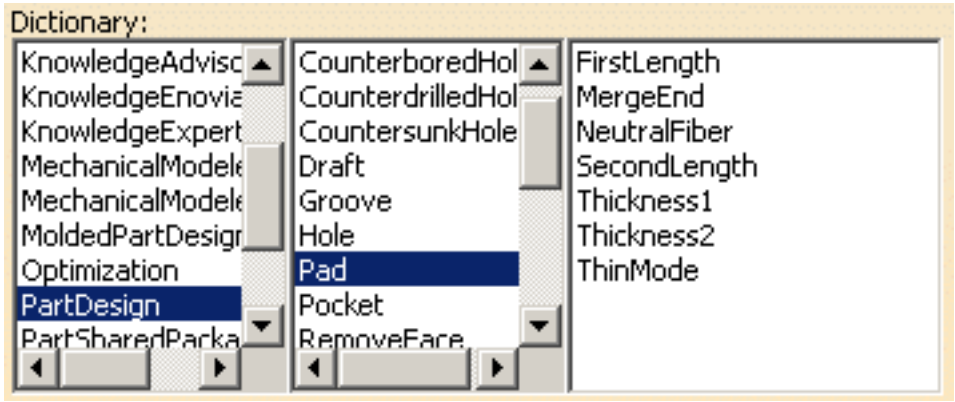
To specify a hole within your script, you have to use one of the holes listed below. Hole is the father type and cannot be used.

- Counterbored Hole
- Countersunk Hole
- Counterdrilled Hole
- Tapered Hole

Pad

Definition:

A pad is a feature created by extruding a sketch.



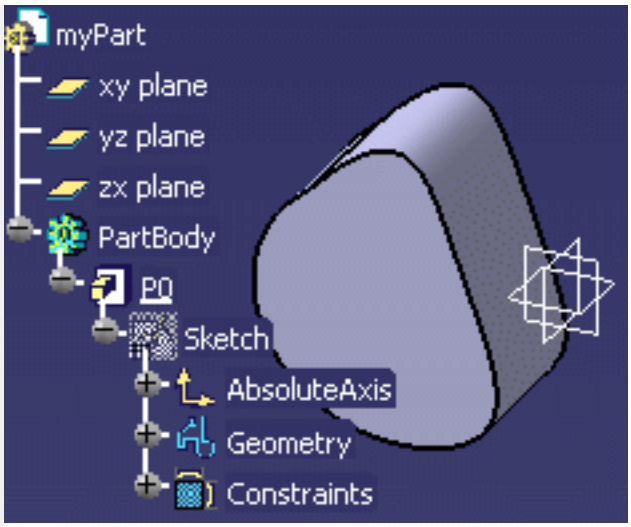
Attributes:

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*).

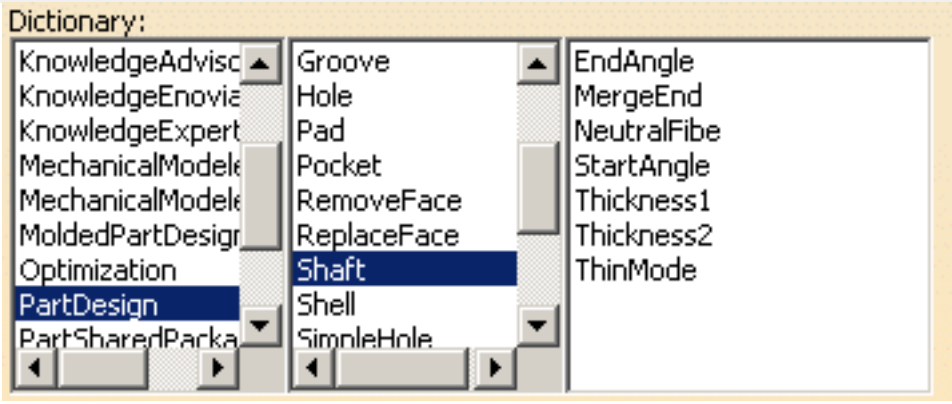
A limit which is not specified is set by default to zero.

```
// 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.



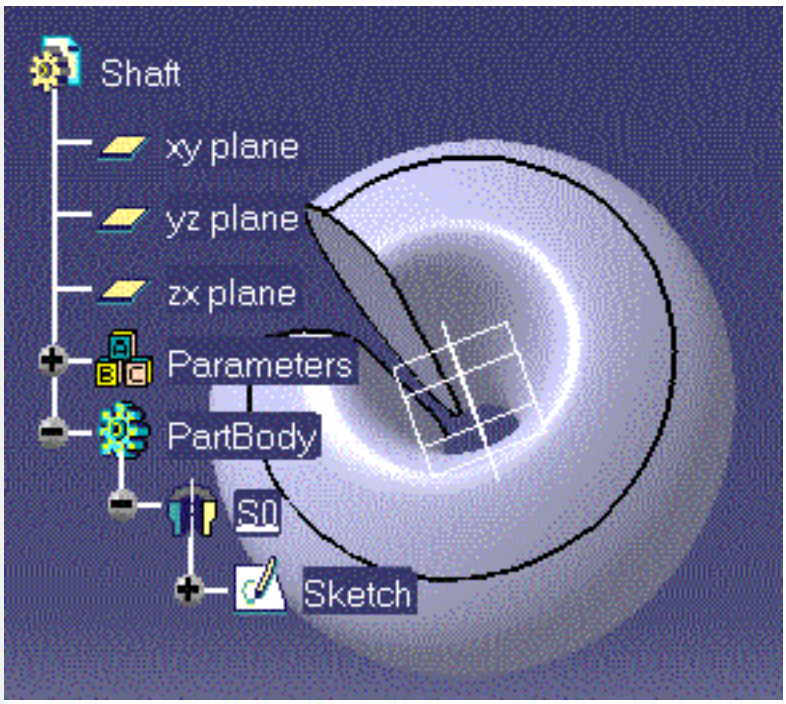
Attributes:

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.

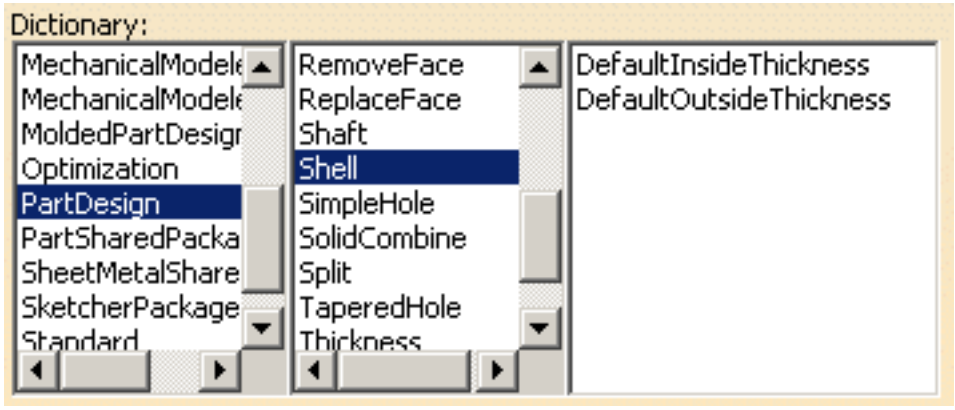
```
/* Use the Insert File Path command from the
contextual menu */
/* to specify the sketch to be imported */
import sketch to be imported. The file must contain
Sketch.1 */
import //Use the Insert File Path command to insert the
Pktsketch_shaft.CATPart file.
MyShaft isa CATPart
 {
  myPart isa Part
   {
    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:

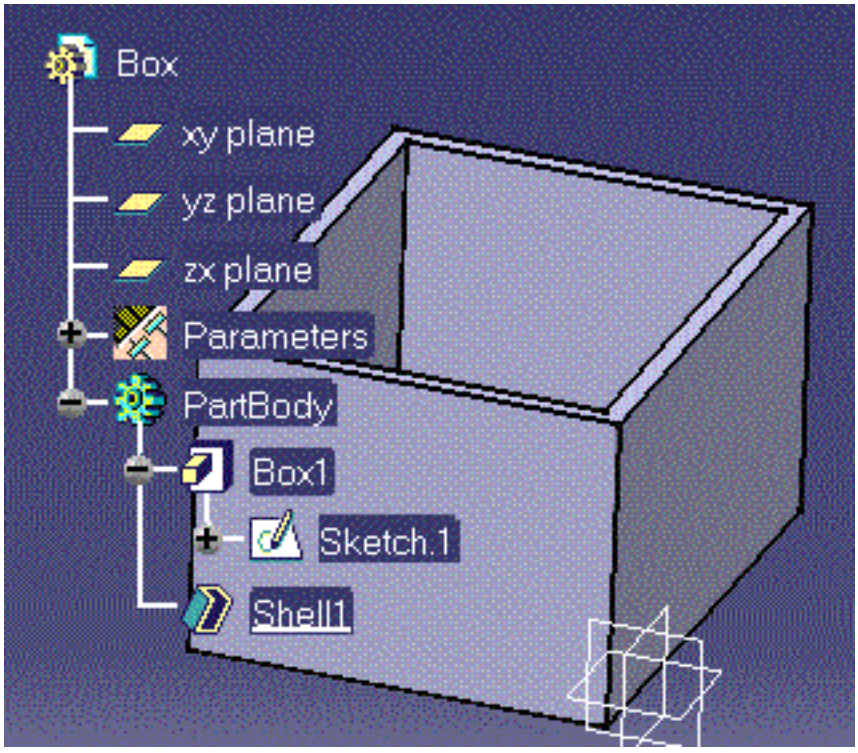
1. create a Shell by using the isa function

   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;
       }
     }
   }
}
```



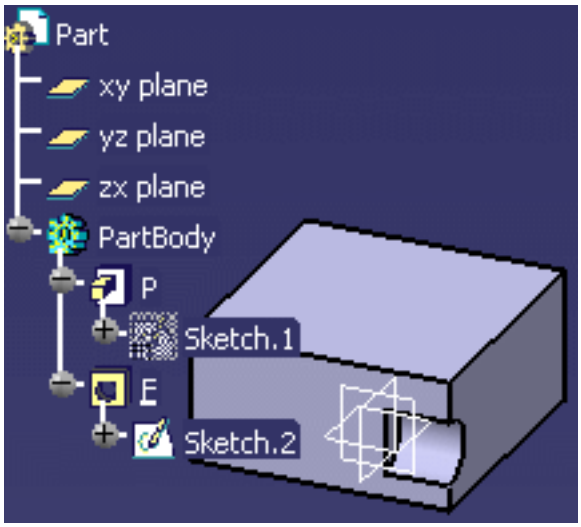
SimpleHole

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:

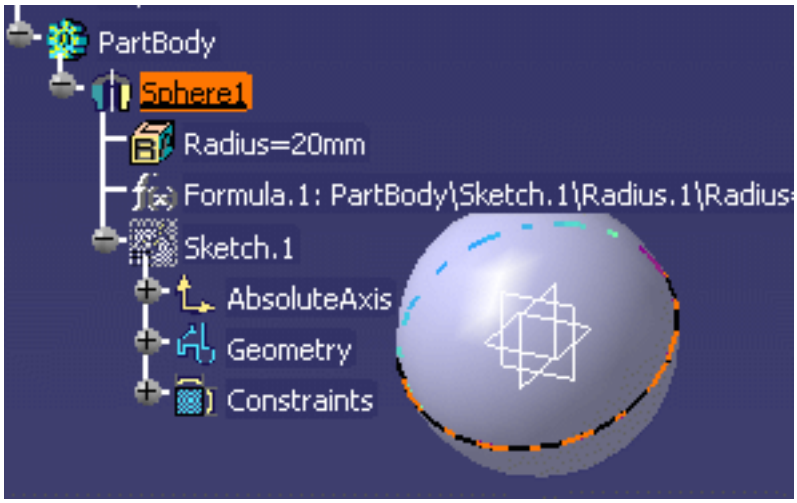
A sphere is a shaft created by rotating half a circle around an axis passing through the arc extremities. The only property is the Radius.

Attributes:

A sphere is defined by the following attribute:

- Radius. The syntax to be used is: **Radius = 20.0 mm**.

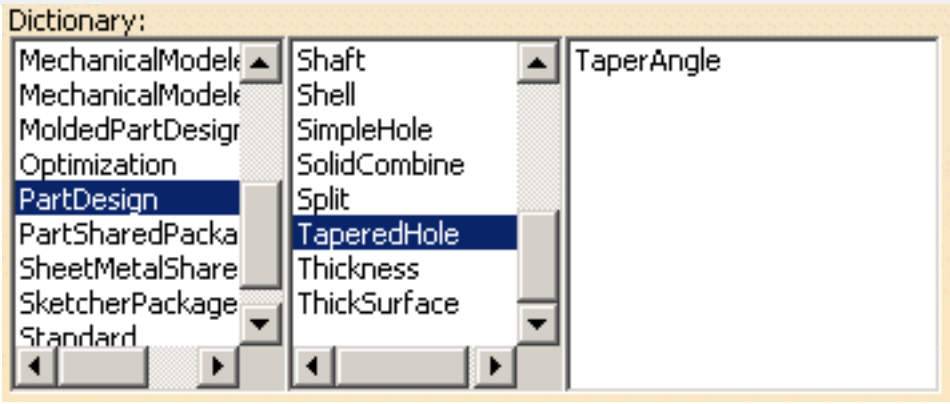
```
MySphere  isa CATPart
 {
  SpherePart isa Part
   {
    PartBody isa BodyFeature
     {
      Sphere1 isa Sphere
       {
        Radius = 20.0 mm ;
       }
     }
   }
}
```



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

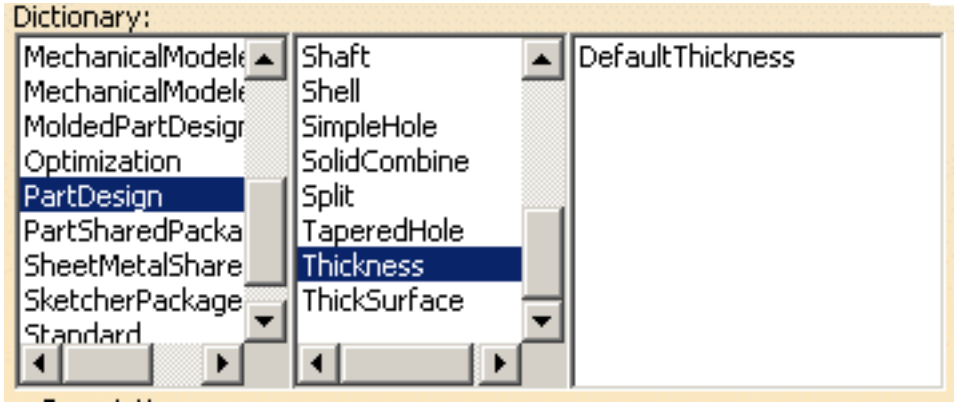
A tapered hole is defined by the following attribute:

- TaperAngle:

Thickness

Definition:

A thick



Attributes

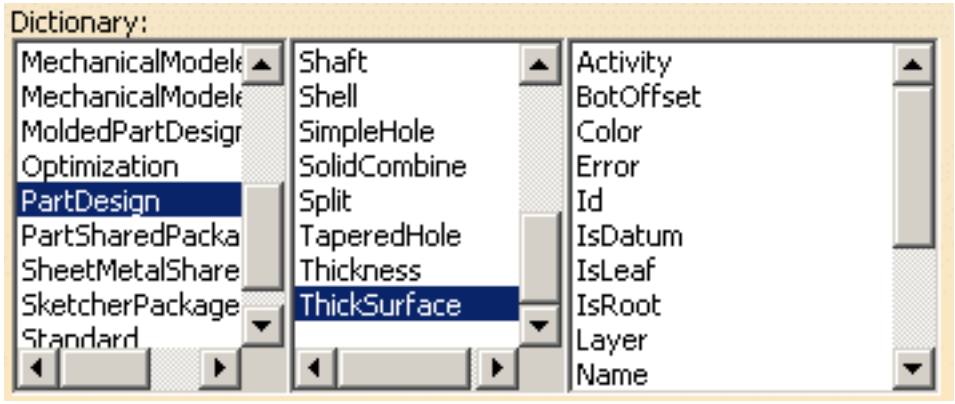
A thickness is defined by the following attribute:

- DefaultThickness:

ThickSurface

Definition:

A thicksurface is a surface to which material was added in two opposite directions.

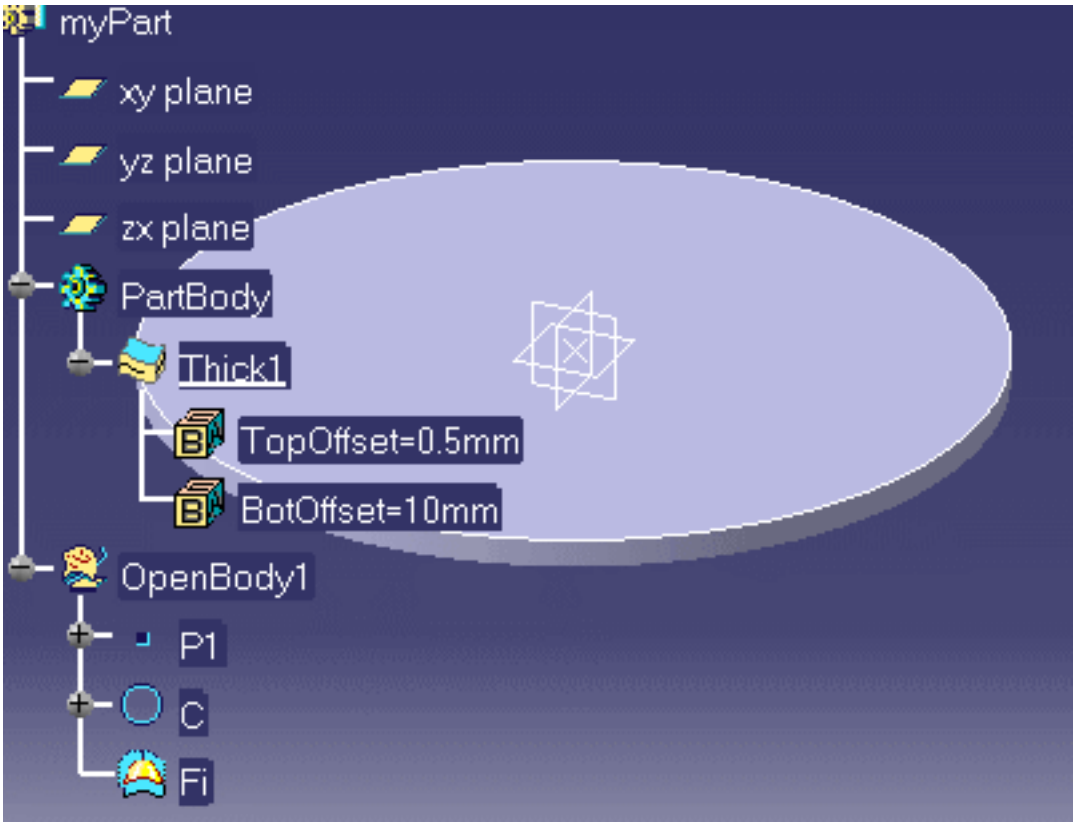


Attributes:

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:

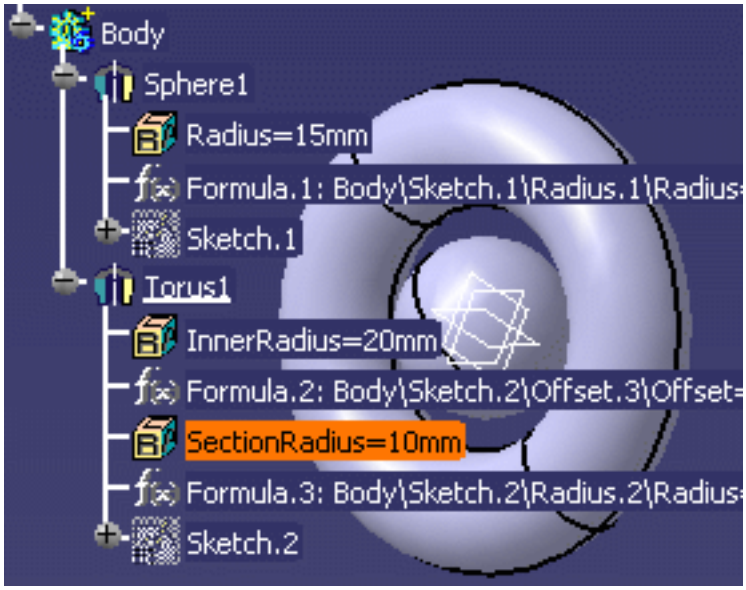
A torus is a shaft created by rotating a circular sketch around an axis.

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
Fillet
Pattern

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

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:

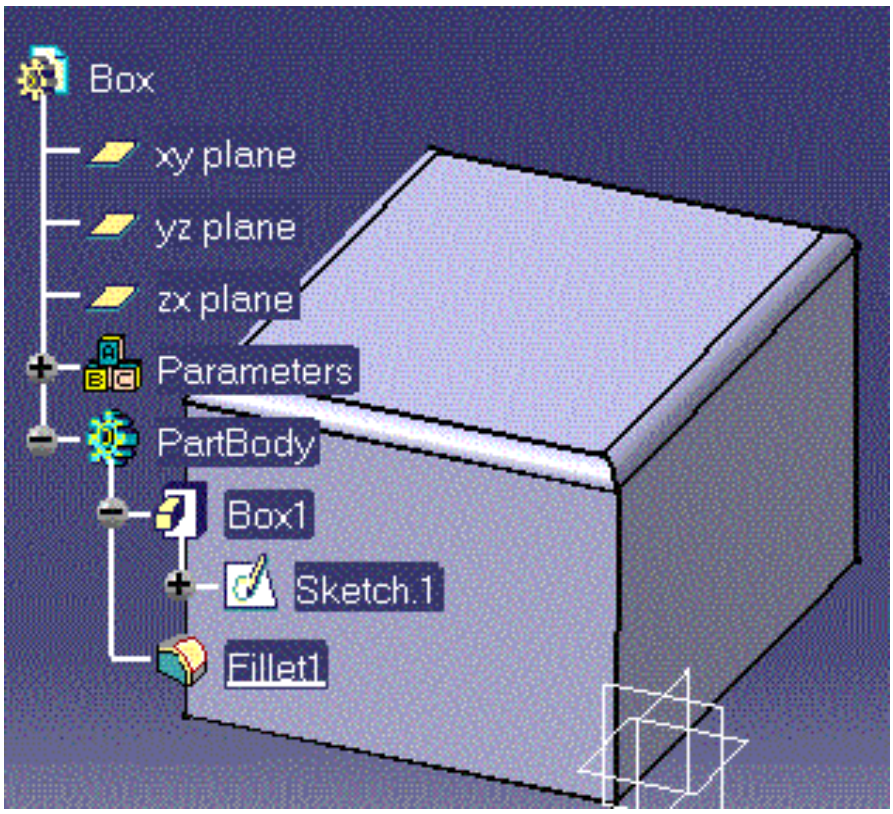
1. Create a Fillet by using the isa keyword.

   ```
   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;
           }
        }
     }
  }
```



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

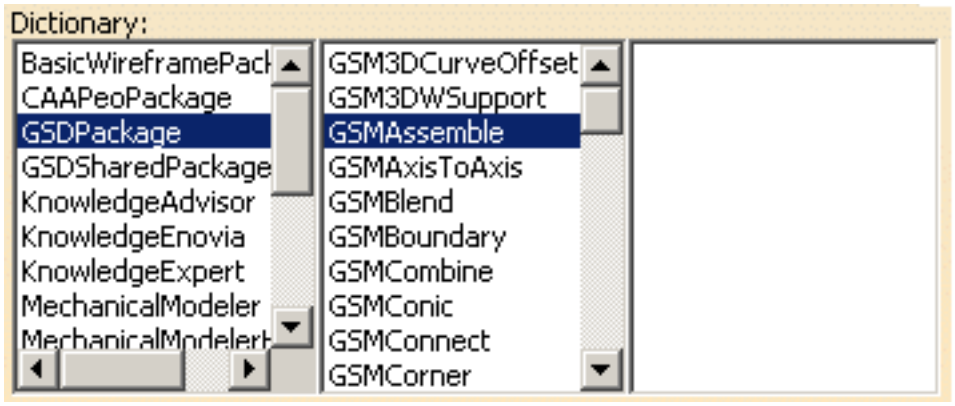
Please find below a table listing the types contained in the GSD package:

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

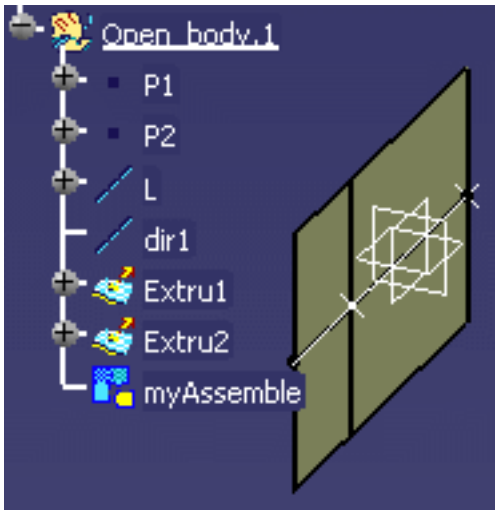
Definition:

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.

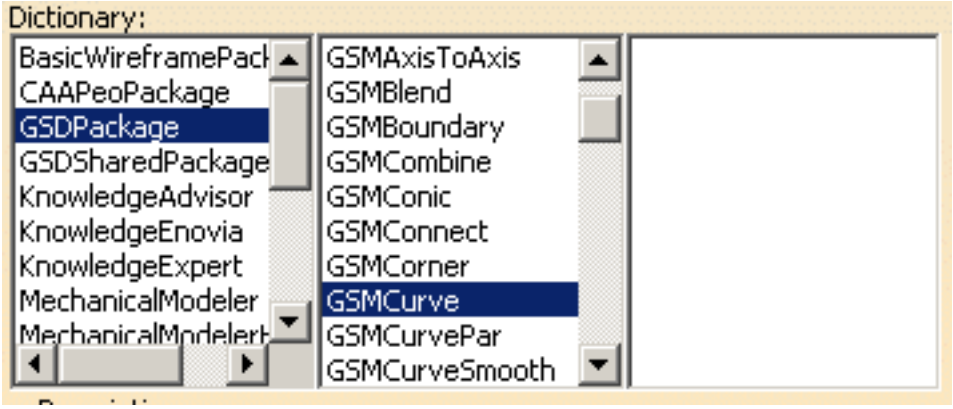


GSMCurve

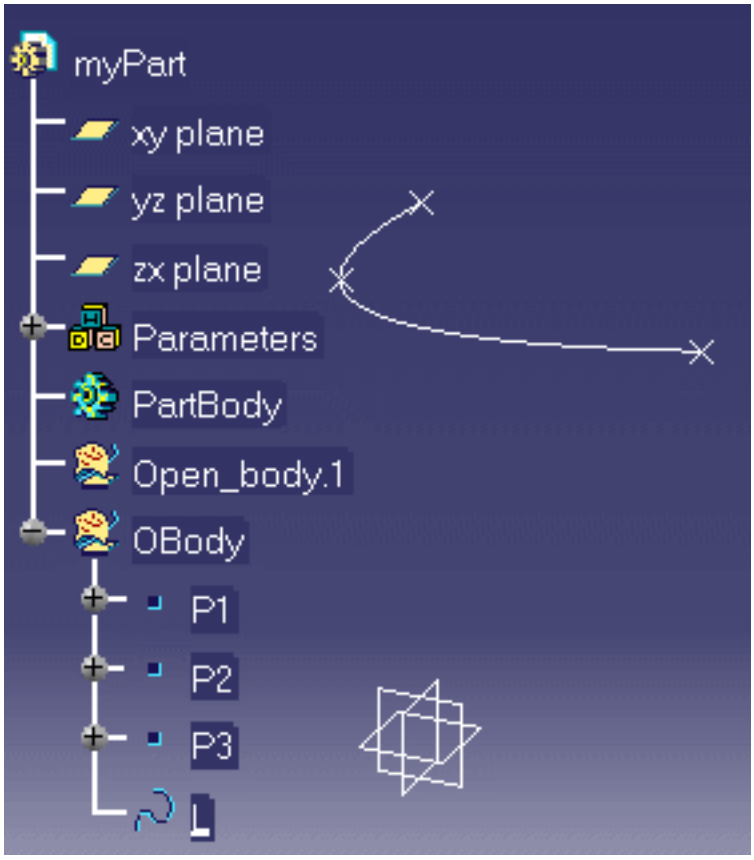
Definition:

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.



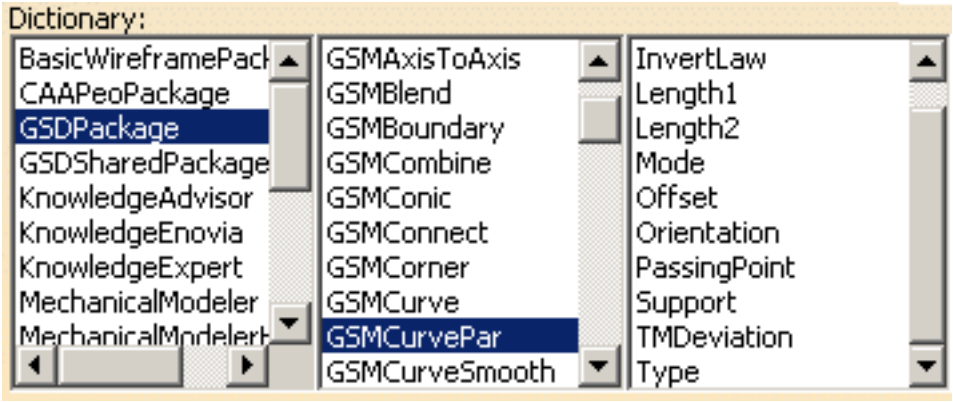
Click here to open the GSMCurveScript script file.



GSMCurvePar

Definition:

An GSMCurvePar object is a Generative Shape Design parallel curve.

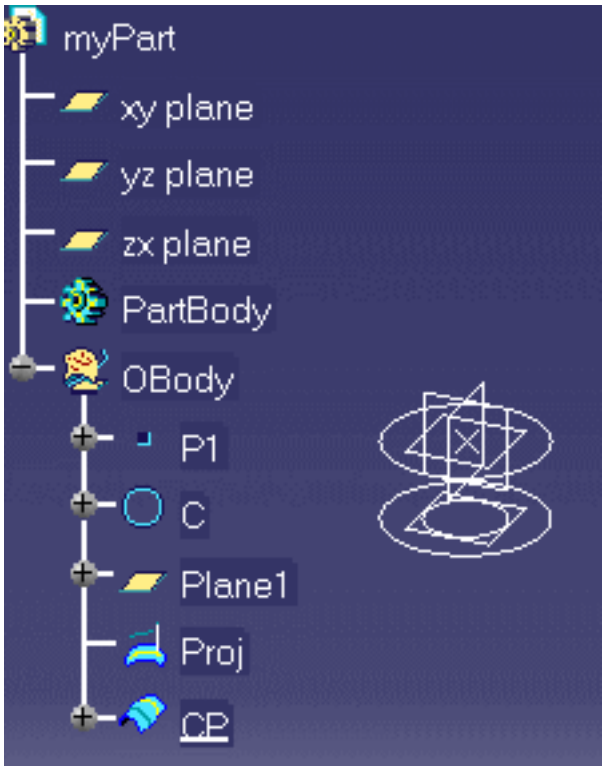


Attributes:

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.

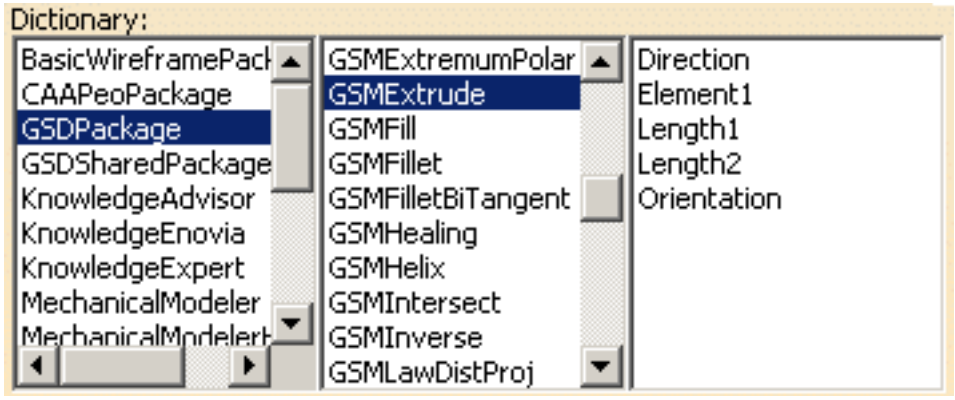


GSMExtrude

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.

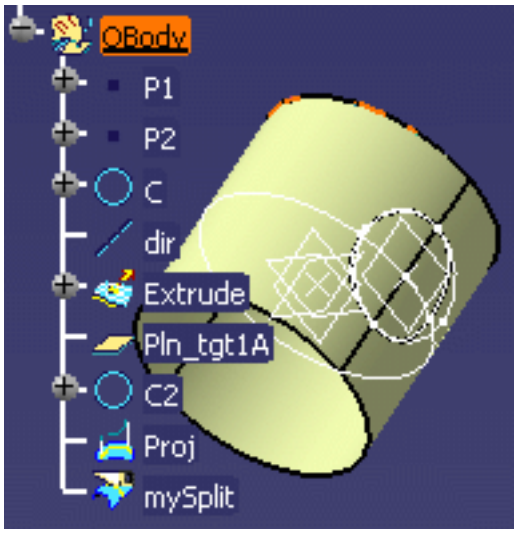


Attributes:

A GSMExtrude is defined by the following attributes:

- *Direction*.
- *Element1*.
- *Length1*.
- *Length2*.
- *Orientation*.

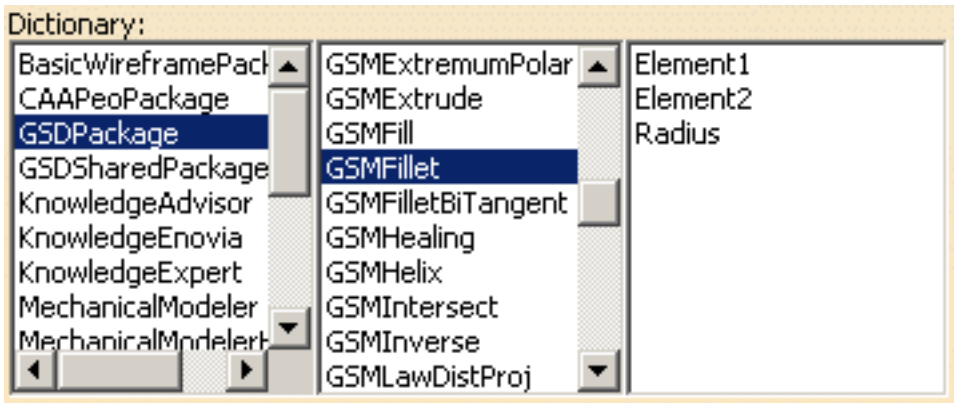
Click here to open the GSMSplit script file.



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.

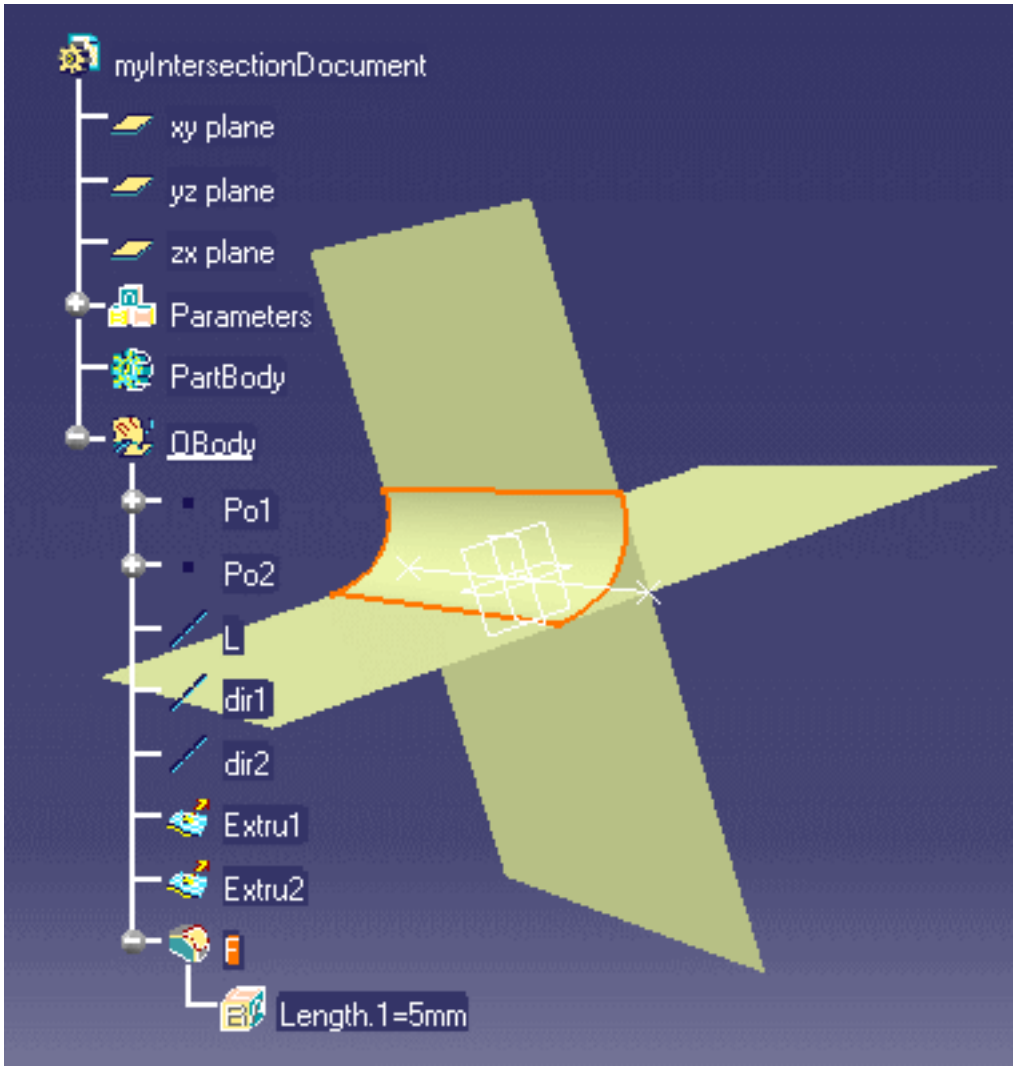


Attributes:

A GSMFillet is defined by the following attributes:

- *Element1*.
- *Element2*.
- *Radius*.

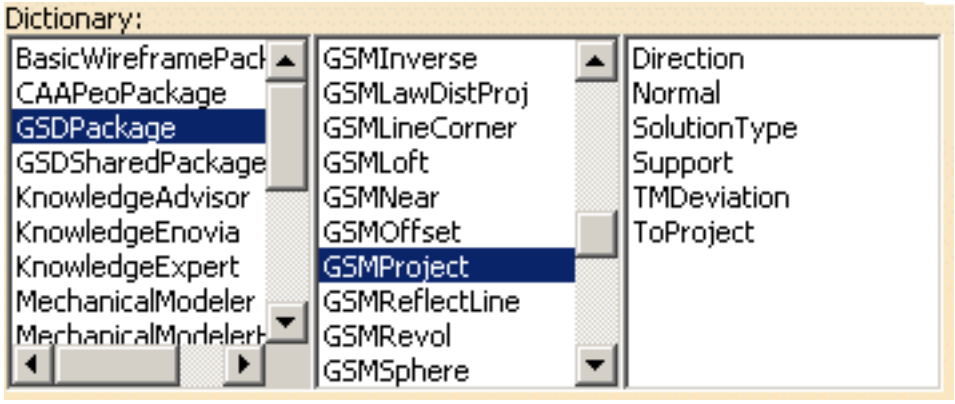
Click here to open the GSMFilletScript.CATGScript file.



GSMProject

Definition:

A Generative Shape Design projection. See the *Generative Shape Design User's Guide* for more information.

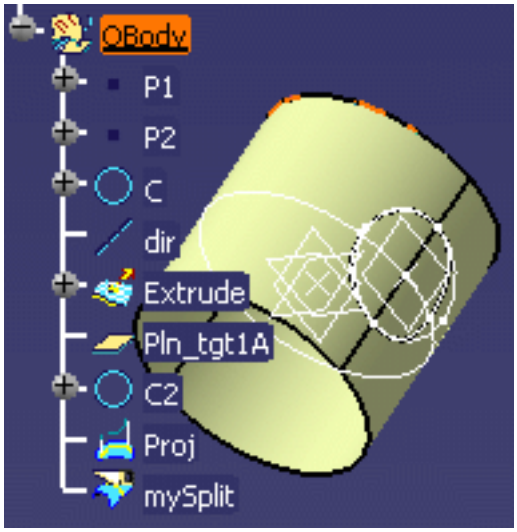


Attributes:

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.

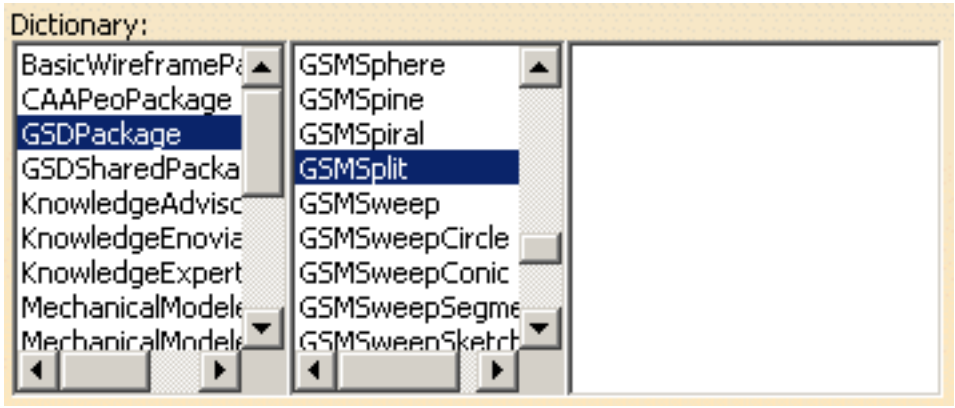


GSMSplit

Definition:

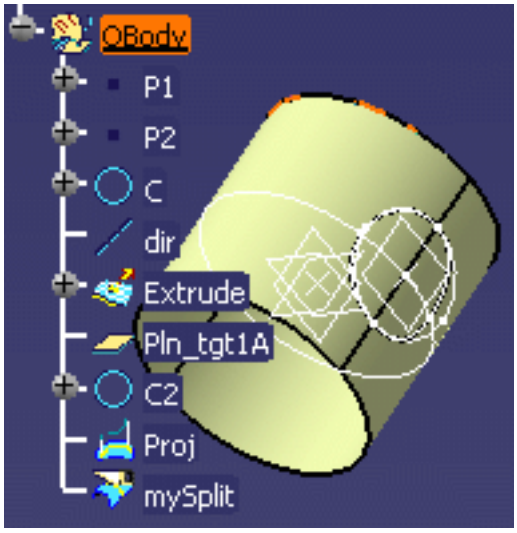
A surface or wireframe element that was split by means of a cutting element. You can split:

- a wireframe element by a point, another wireframe element or a surface
- a surface by a wireframe element or another surface.



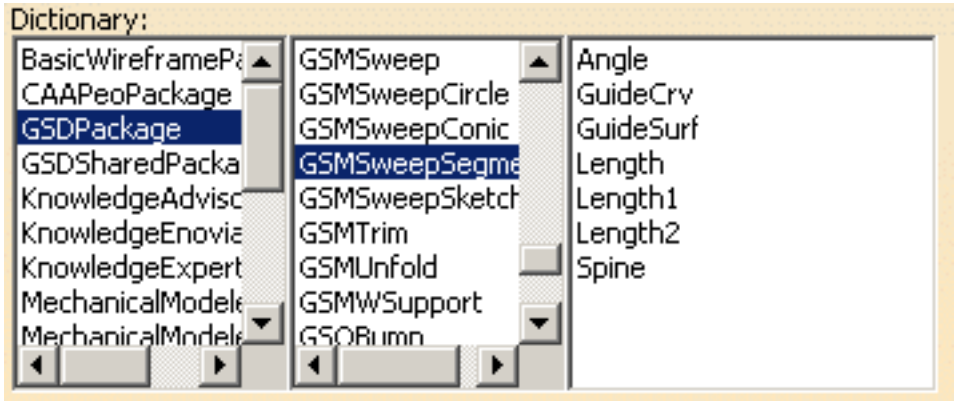
You can split geometry by clicking the **Split** icon () in the Generative Shape Design workbench.

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

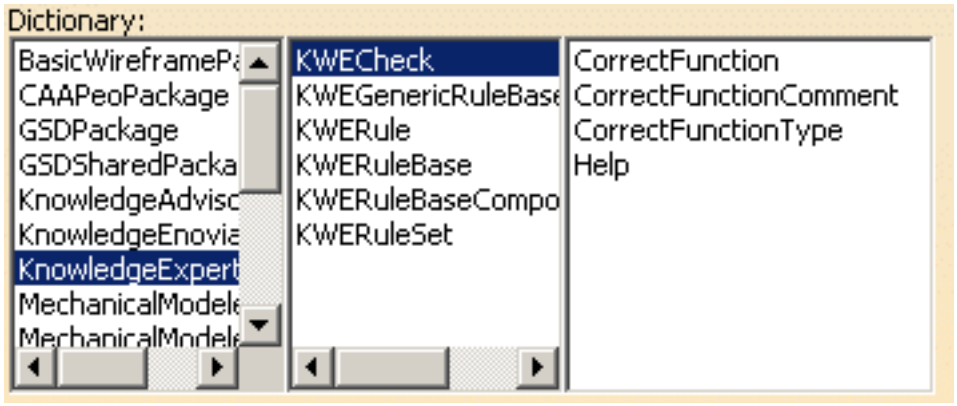
Please find below a table listing the types contained in the Knowledge Expert package:

KWECheck	*KWEGenericRuleBaseComponent*	KWERule
KWERuleBase	KWERuleSet	*KWERuleBaseComponent*

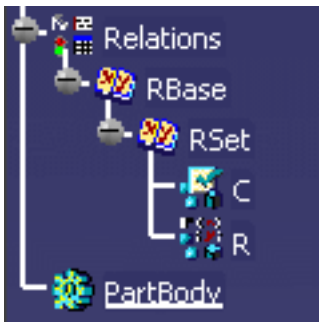
KWECheck

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.



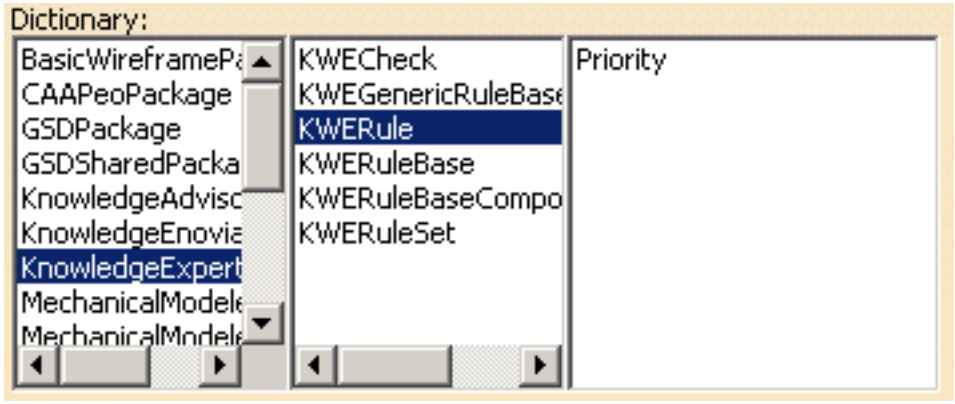
Click here to open the KnowledgeExpertScript file.



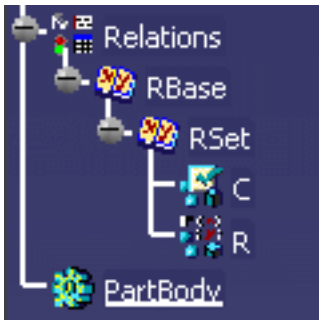
KWERule

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.



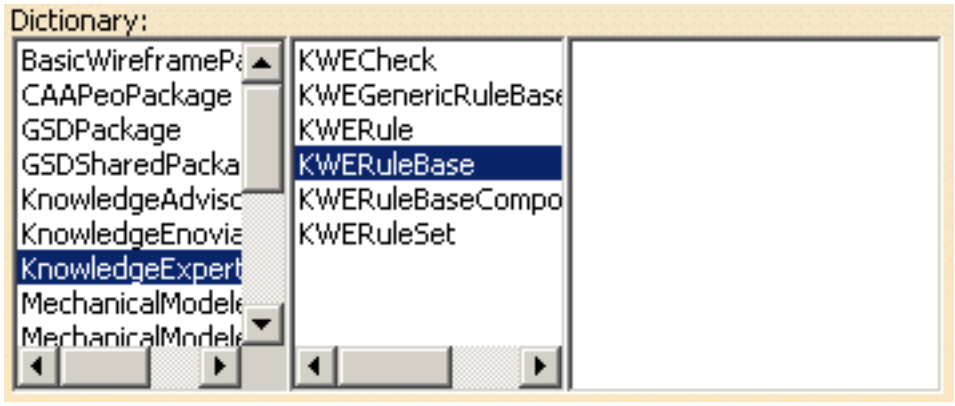
Click here to open the KnowledgeExpertScript file.



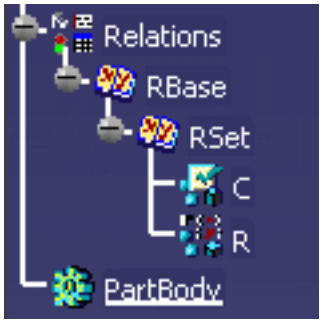
KWERuleBase

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.



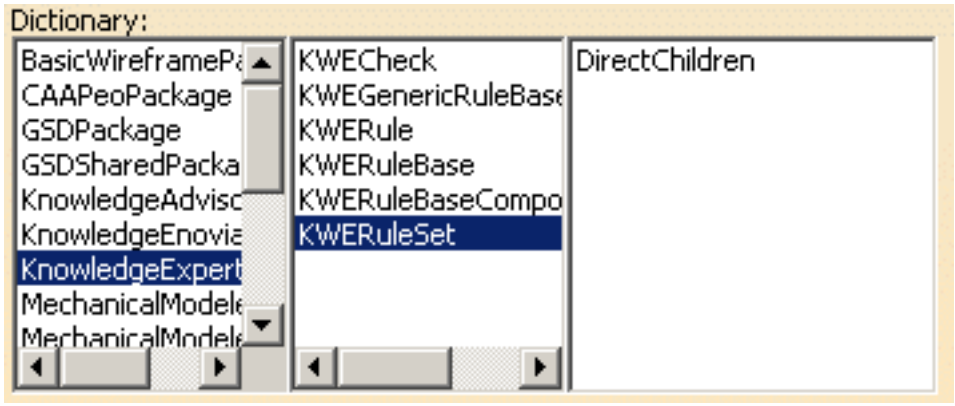
Click here to open the KnowledgeExpertScript file.



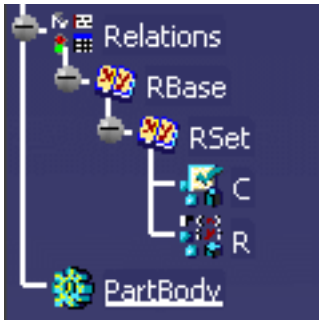
KWERuleSet

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

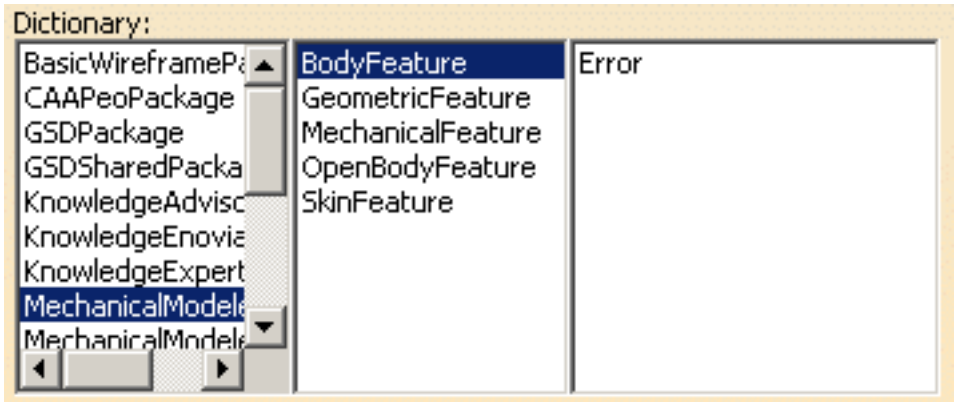
Some types and attributes were changed. Please find below a conversion table listing the old types, their attributes, their new names (if any) as well as their attributes:

BodyFeature	*GeometryFeature*
MechanicalFeature	*OpenBodyFeature*
OpenBodyFeature	

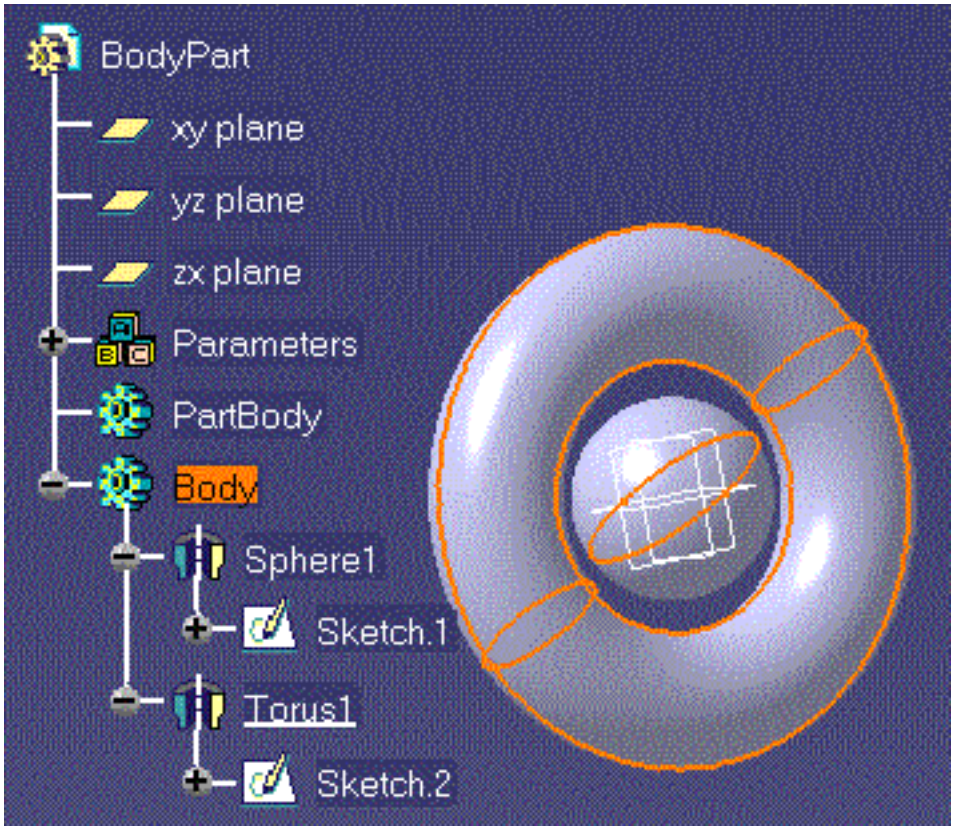
BodyFeature

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 ;
           }
        }
     }
  }
```



Product Knowledge Template Interoperability

In this section, you will find information about the interoperability between *CATIA* Product Knowledge Template and other applications listed below:

ENOVIA VPM V5 Interoperability
ENOVIAVPM Interoperability

ENOVIA VPM V5 Interoperability

Optimal CATIA PLM Usability for Product Knowledge Template
Working with Assembly Templates in ENOVIA VPM V5
Saving and Using Assembly Templates in ENOVIA VPM V5
Working with Assembly Templates in ENOVIA LCA Using VPM Navigator
Instantiating a Part Template From ENOVIA VPM V5 Using the Document Chooser

Optimal CATIA PLM Usability for Product Knowledge Template

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). Product Knowledge Template (PKT) has been configured to work in the Workpackage mode.

Product Knowledge Template Commands in ENOVIA VPM V5

Please find below the list of the Product Knowledge Template commands along with their accessibility status in ENOVIA VPM V5.

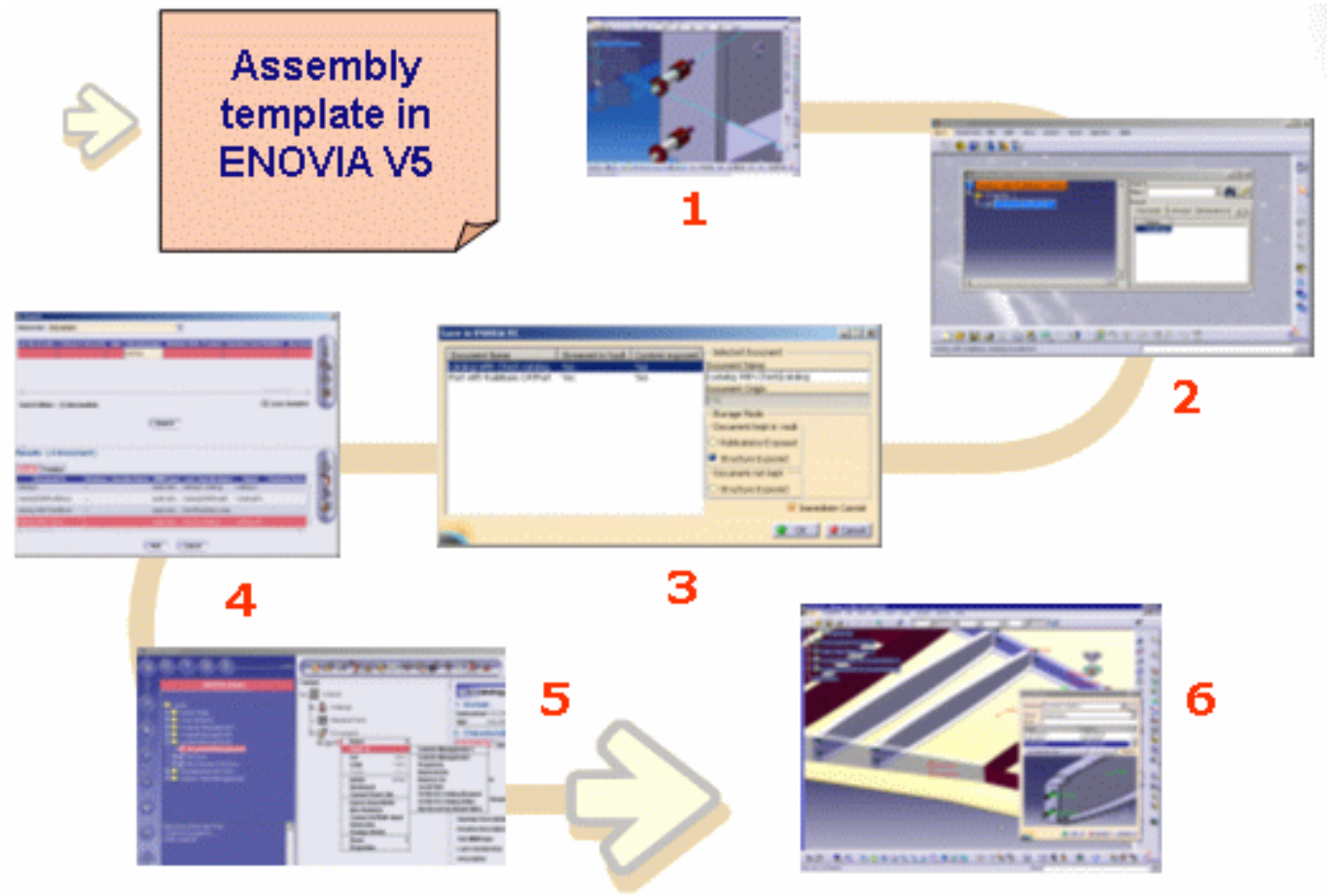
Commands	Accessibility in ENOVIA VPM V5	Comments
Create a Generative Script	Available	None
Create a PowerCopy	Available	None
Create a UserFeature	Available	None
Create a Document Template	See Working with Assembly Templates in ENOVIA.	None
Save in catalog	Available	None
Instantiate From Selection	Available	None
Instantiate From Document	Available	None
Open Catalog	Available	None

To ensure seamless integration, you must have both a *CATIA* and ENOVIA session running.

Working with Assembly Templates in ENOVIA VPM V5

Note that the methodology described in this topic also applies to ENOVIAVPM.

Saving an Assembly Template in ENOVIA VPM V5



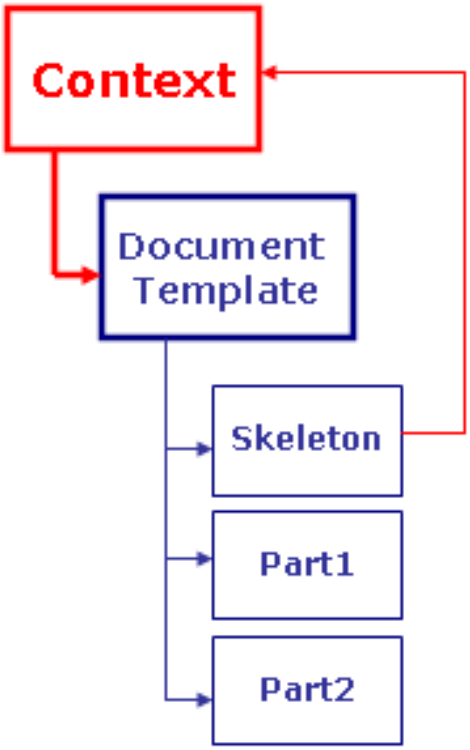
To save an assembly template in ENOVIA VPM V5, the user first creates the assembly and saves it in ENOVIA VPM V5. Then, he creates the document template and saves the product in ENOVIA VPM V5 (**1**). He creates a CATIA catalog document referencing this assembly template (**2**), and saves the catalog as a new document (Document kept mode) in ENOVIA VPM V5 (**3**). The catalog becomes available in the Search window in ENOVIA VPM V5 (**4**). He sends the catalog from ENOVIA VPM V5 to *CATIA* (**5**) and instantiates the assembly template in *CATIA* (**6**).

Methodology

Defining the assembly template

- The assembly template must be stored in a product saved using the document kept storage mode (work package).
- The assembly template may contain internal relational design links: The context of those links must be the assembly root (or one of its sub-CATProducts) except in the case of a skeleton that can be contextual outside the context.

- Links to external documents stored in ENOVIA VPM V5 can be created if they are loaded in session.



- The recommended methodology consists in:
  - Creating a skeleton containing isolated geometry (not contextual). This skeleton is designed in context.
  - Designing the rest of the model in context of the document template.

Part1 and Part2 can be contextual to the skeleton via the Document Template product.

The skeleton can be contextual. In this case, the inputs will be automatically valuated at instantiation. If not, the user will have to select them.

Note that if the document template was designed in context, the documents located above this template in the specification tree will be required at instantiation.

Referencing the assembly template in a catalog

- The catalog must be stored using the Document kept storage mode.

Instantiating the assembly template

- The assembly template can be instantiated into a product saved using the Document Kept storage mode (scenario 2) but also in a product explode (Document not kept) (see scenario 1).
- When instantiating in an explode context, the option **Keep link with selected object** should not be selected: Relational design links cannot be saved in the exploded product. An error message will be raised if the user tries to instantiate a document template with this option activated. The assembly instantiated is stored as a work package.
- The result of an instantiation cannot be saved in Structure Exposed storage mode.
- If the destination product was saved in Structure Exposed storage mode, or if it is located below a product saved in Structure Exposed storage mode, the instantiation will not be possible if the template contains contextual parts whose context is outside the template but whose external references point items located in the template.
- Managing reference-to-reference link/Automatic input: if one of the link relies on a reference-to-reference basis, a reference-to-reference link is created at instantiation if the **Keep link with selected object** option is checked.

- When the destination assembly contains at least one Product in Explode mode, the context is the first work package found. If the instantiation occurs at the work package level, the context will be the work package and the user will only be able to select the items located below this work package (if the **Keep link with selected**

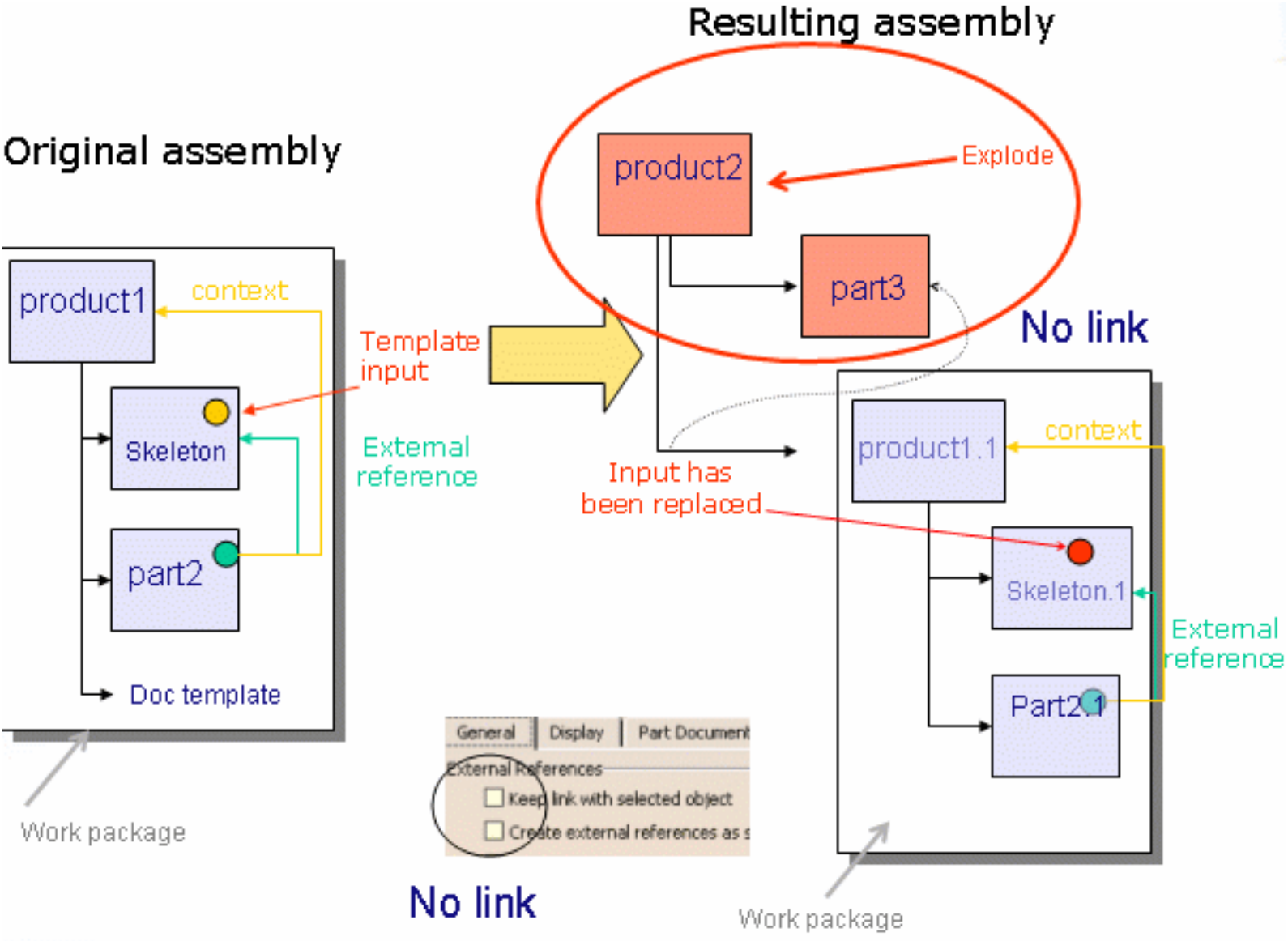
object option is checked).

- When the **Use root context in assembly** option is checked, the behavior is the default document template behavior.

Supported Scenarios

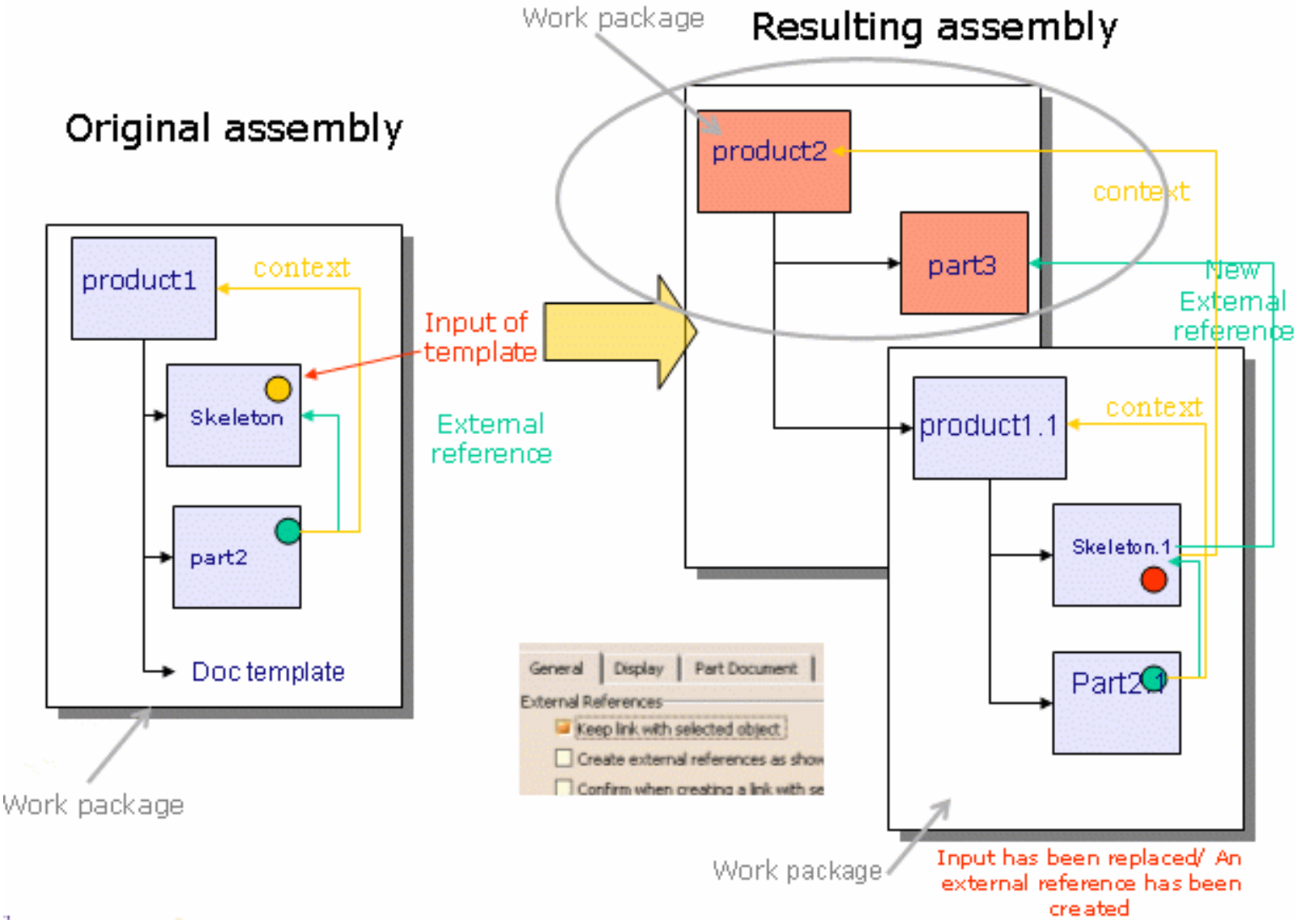
Destination Product: Document not kept mode

In the graphic below, the assembly template has one input and the assembly contains contextual links. If the document contains contextual links, the context must be the root assembly. The instantiation can occur in a product saved in Publications Exposed storage mode. It is not possible to instantiate an assembly template into a product saved in Structure Exposed storage mode if this instantiation is going to create contextual links, that is to say if the **Keep link with selected object** check box option is checked in the **Tools->Options** menu and if the template has inputs.



Destination Product: Document kept Scenario

In the graphic below, the assembly template has one input and the assembly contains contextual links. The **Keep link with selected object** option is supported.

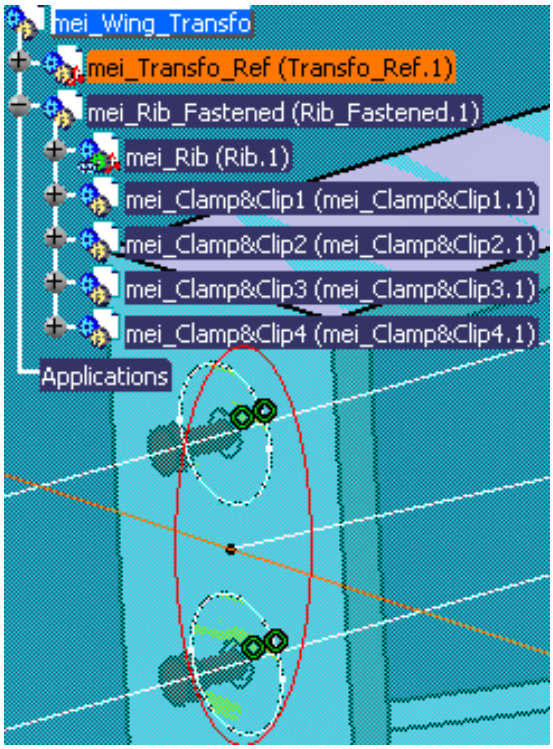


Saving and Using Assembly Templates in ENOVIA VPM V5

This task is designed to show the user how to save assembly templates in ENOVIA VPM V5 and how to re-use them in CATIA . In the scenario described below, the user instantiates ribs into a plane wing using the document template feature.

The scenario is divided into the following steps:

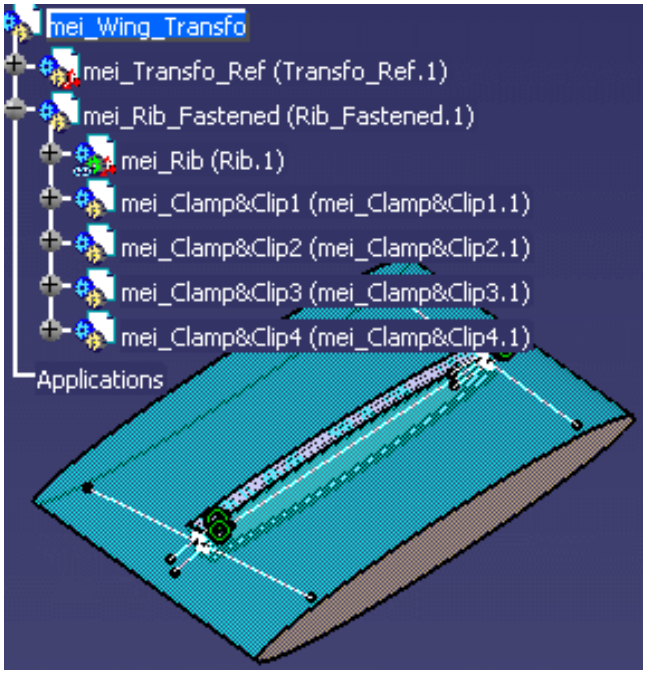
- The user opens the Wing_Transfo.CATProduct file. This file already contains 4 instantiated clamping pins (see picture opposite). He saves it in ENOVIA VPM V5.
- The user creates the document template in the Rib_Fastened.CATProduct file and saves the file in ENOVIA VPM V5. Then he creates a catalog (Rib_Template.catalog) referencing the document template and saves the catalog in ENOVIA VPM V5.
- The user saves the Left_Wing.CATProduct file in ENOVIA VPM V5 and reloads it in CATIA. Then he loads the Rib_Template.catalog in CATIA and instantiates the document template into the wing.



Before you start, make sure you have checked the **Keep link with selected object** option in the **Part Infrastructure->General** tab.

Saving the Wing_Transfo.CATProduct file in ENOVIA VPM V5

1. In CATIA, open the Wing_Transfo.CATProduct file. The following image displays:

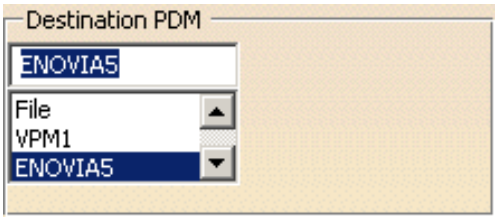


2. Save your files in ENOVIA VPM V5. To do so, proceed as follows:

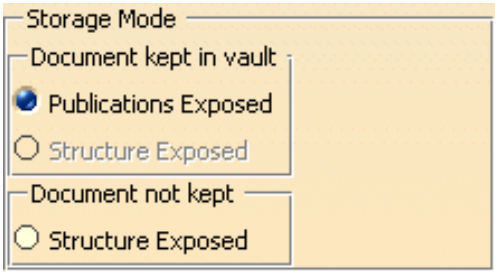
- In CATIA, click the **Connect to ENOVIA LCA** icon ( ) to connect your ENOVIA VPM V5 database.

- Click the **Set PDM Properties** icon ( ).

  - In the Set PDM Properties window, select **Enovia5** in the Destination PDM scrolling list.



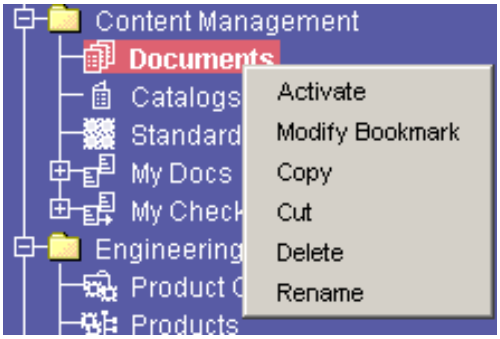
  - Select the **Publications Exposed** storage mode. Click **OK** when done.



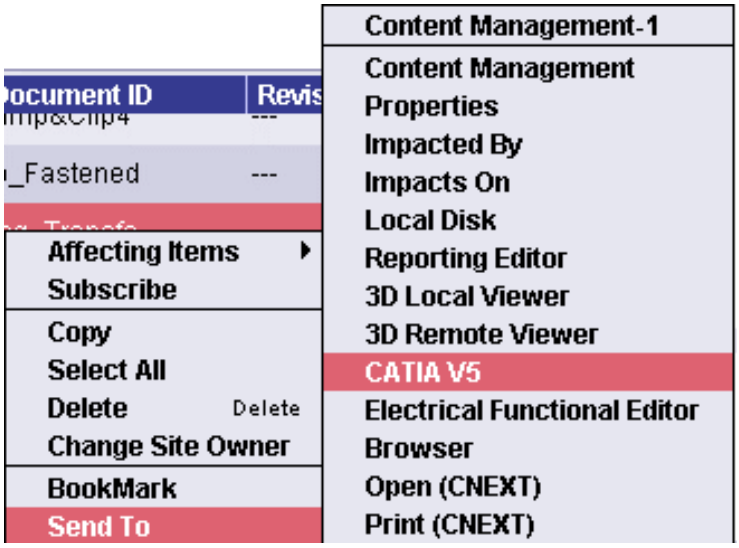
- In CATIA, click the **Save data in ENOVIA LCA Server...** icon ( ). The Save in ENOVIA V5 dialog box displays.

- Check the **Immediate Commit** check box (if need be) and click **OK**. Your data are saved in the ENOVIA VPM V5 database. Close the file in *CATIA*.

3. In ENOVIA VPM V5, click the **ENOVIA Home** icon ( ). Expand the Content Management node, right-click the Documents folder and select the **Activate** command.



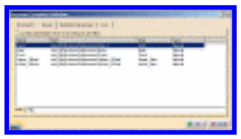
4. The Content Management Startup Selection dialog box displays. Click the **Search** Documents radio button and click **OK** to launch the search.
5. In the opening Search window,
   - select **Document Revision** in the **Search for:** scrolling list.
   - select the **Creator** field, enter the Creator's name and click **Search**.
6. The Wing_Transfo.CATProduct file displays. Right-click it and select the **Send To->CATIA V5** command. The file opens in *CATIA*. Close your file.



Creating the document template in the Rib_Fastened.CATProduct file

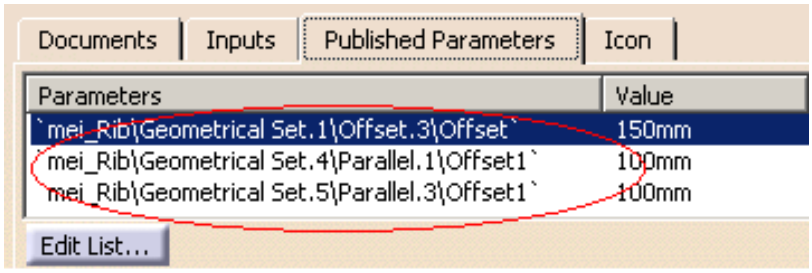
7. Open the Rib_Fastened.CATProduct file.
8. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench.
9. Click the **Create a Document Template** icon ( ). The **Document Template Definition** window displays.
10. In the **Document Template Definition** window, click the **Inputs** tab to select the inputs. In the geometry, expand the External References node located below the Rib node. Select all external references and assign them a role (see table below.)

Name	Role
Line.1	Web
Rear	Rear
Front	Front
Upper Offset	Upper_New
Lower Offset	Lower_New



11. Click the **Published Parameters** tab to publish parameters. To do so, proceed as follows:

    - Click the Edit List... button. The **Select parameters to insert** window displays.



    - Use the arrow key to select the following parameters:
        - Geometrical Set.1\Offset.3\Offset
        - Geometrical Set.4\Parallel.1\Offset1
        - Geometrical Set.5\Parallel.3\Offset1

    - Click **OK** twice. The Document template is added to the KnowledgeTemplates node.

12. Click the **Save data in ENOVIA LCA Server...** icon ( ) to save your document in ENOVIA VPM V5. Do not close your file in *CATIA*.

Creating and saving the catalog containing the document template

13. From the **Start->Infrastructure** menu, access the **Catalog Editor** workbench.

14. Click the **Add Family** icon ( ). In the Component Family Definition dialog box, enter Rib and click **OK** when done.

15. Double-click the Rib family and click the **Add Component** icon ( ). Click the **Select external feature** button and select the document template in the Rib_Fastened.CATProduct file.

16. Click the **Set PDM Properties** icon ( ). In the Set PDM Properties window, select **Enovia5** in the Destination PDM scrolling list, and select the **Publications Exposed** storage mode. Click **OK** when done.

17. Click the **Save data in ENOVIA LCA Server...** icon ( ). The Save in ENOVIA V5 dialog box displays.

18. Check the **Immediate Commit** check box (if need be) and click **OK**. Your catalog is saved in the ENOVIA VPM V5 database. Close the files in *CATIA*.

Saving the Left_Wing.CATProduct file in ENOVIA VPM V5

19. In CATIA, open the Left_wing.CATProduct file.

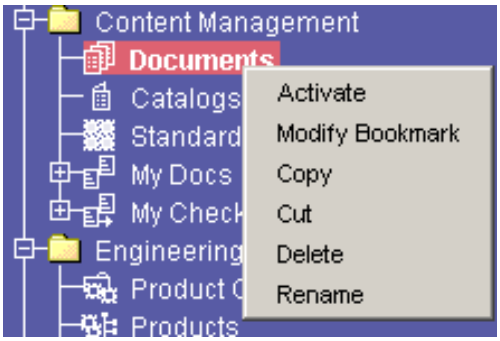
20. Save your files in ENOVIA VPM V5. To do so, proceed as follows:

    - Click the **Set PDM Properties** icon ( ). In the Set PDM Properties window, select **Enovia5** in the Destination PDM scrolling list, and select the **Publications Exposed** storage mode. Click **OK** when done.
    - In CATIA, click the **Save data in ENOVIA LCA Server...** icon ( ). The Save in ENOVIA V5 dialog box displays. Check the **Immediate Commit** check box (if need be) and click **OK**.
    - Your data are saved in the ENOVIA VPM V5 database.

Instantiating the document template into the Left_Wing.CATProduct file

21. In ENOVIA VPM V5, right-click the saved product and select the **Send To->CATIA V5** command. The document opens in CATIA.

22. In ENOVIA VPM V5, click the **ENOVIA Home** icon ( ). Expand the Content Management node, right-click the Documents folder, and select the **Activate** command.



23. The Content Management Startup Selection dialog box displays. Click the **Search** Documents radio button and click **OK** to launch the search.

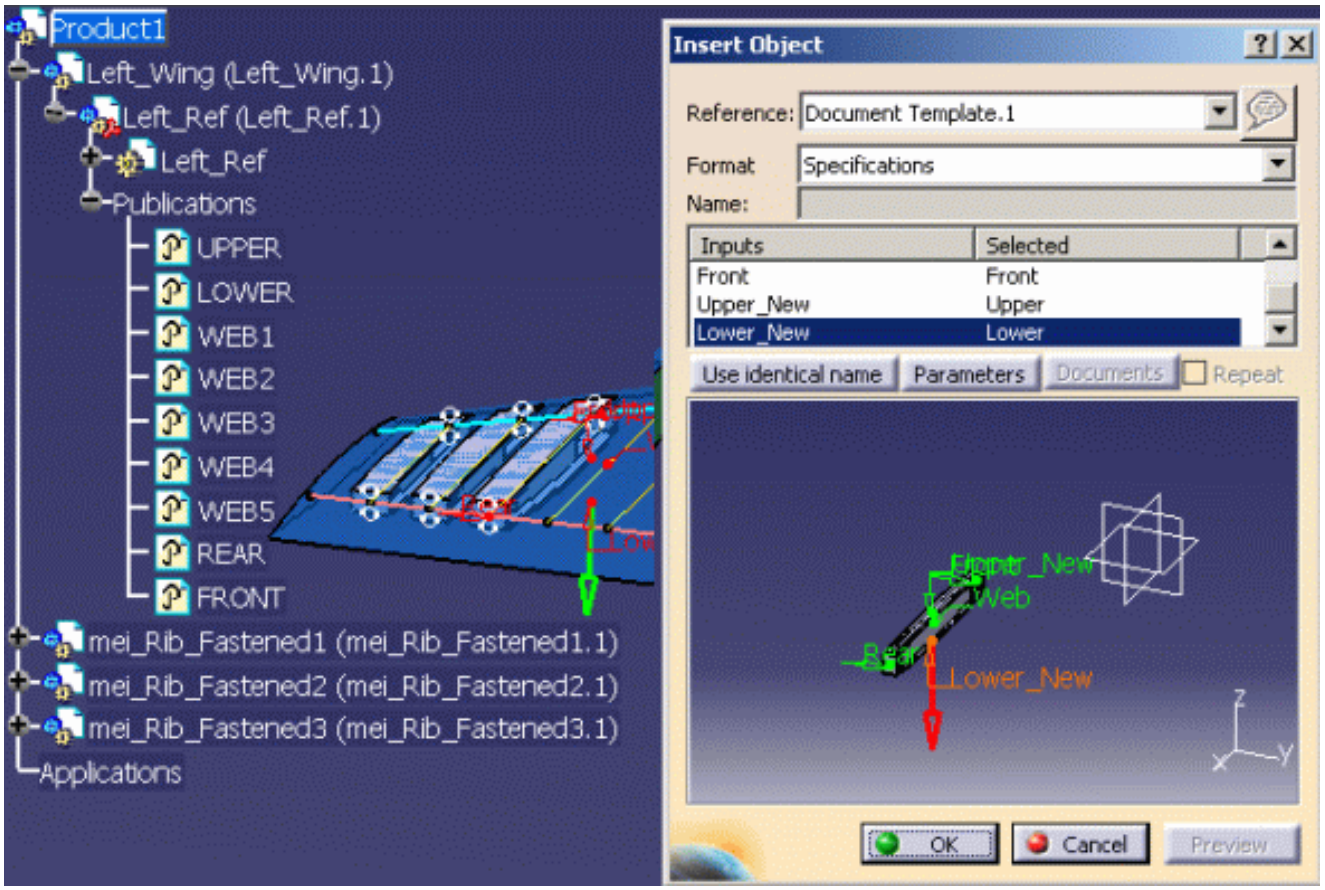
24. In the Search window,

    - select **Document Revision** in the Search for: scrolling list.
    - select the Creator field, enter the creator's name and click **Search**.

25. In the list, right-click the Rib_Template.catalog file, and select the **Send To->CATIA V5 Catalog Browser** command.

26. The catalog browser displays in *CATIA*. Double-click Rib, and DocumentTemplate.1.

27. Expand the Left_Wing/Left_Ref Publications node in the specification tree, and instantiate the rib template contained in the catalog using the wing published features. Repeat this step 4 times.



28. In *CATIA*, click the **Save data in ENOVIA LCA Server...** icon (). Your data are saved in ENOVIA VPM V5.

Working with Assembly Templates in ENOVIA VPM V5 Using VPM Navigator

To define assembly templates, external documents can be selected whether in session or by making a query in VPM Navigator.

To instantiate an assembly template, you must have created a catalog and stored it in ENOVIA. Then, perform a query and send the catalog to the catalog browser.

To know more about the VPM Navigator, see the *VPM Navigator User's Guide*.

Instantiating a Part Template From ENOVIA VPM V5 Using the Document Chooser

This task explains how to instantiate a part template stored in ENOVIA VPM V5 via the Document Chooser.

- Make sure you have selected **ENOVIAV5** in the **View -> Toolbars** menu.
- To carry out this scenario, make sure you have enabled **ENOVIA** in the Document Environments field (**Tools->Options->General->Document**.)

It is now possible to associate non CATIA (ENOVIA VPM V5, ...) documents to a template. To do so, make sure you have enabled the desired environment in the Document Environments field (**Tools->Options->General->Document**.) Your documents will be accessed via the Documents Chooser.

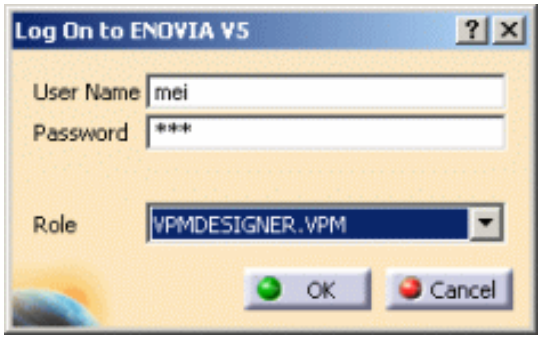
To carry out this scenario, you will need the following documents:

- BottomCase.CATPart
- Battery.CATPart
- Electronic.CATProduct
- PlanarCard.CATProduct
- InteractiveBoard.CATPart
- ChipAC110.CATPart
- Chip_AC20.CATPart
- Chip_AC30.CATPart
- Capacitor_500.CATPart
- Capacitor_700.CATPart
- Speaker.CATPart
- Body.CATPart
- Lens.CATPart
- Indus.CATPart
- LCD30-28.CATPart
- FrontShell.CATPart
- Main_Shape.CATPart

1. Create a Product in ENOVIA called MobilePhone. Note that you can close ENOVIA VPM V5 after creating the product.

Sending the product to CATIA

2. To send the MobilePhone product to CATIA, proceed as follows:

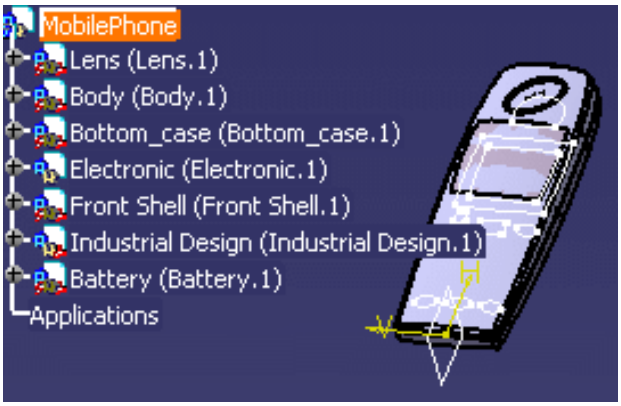


- In CATIA, click the **Connect** icon () in the **ENOVIA V5** toolbar. The Log On to dialog box displays.
- In the Log On to ENOVIA V5 dialog box, enter your **User Name**, your **Password** as well as your **Role**, and click **OK**.

- Click the **Search ENOVIA** icon (). The VPM Search dialog box displays.
- In the Objects scrolling list, select the **Product** option.
- If need be, enter the Owner's ID.
- Click **OK** when done. The Product ID displays in the Result window.
- Double-click the MobilePhone product in the Results list.
- Right-click the root product (MobilePhone) and select the **Open...** command. The Open Modes dialog box displays. Click **OK**. The CATIA view displays.

Adding components to the product

3. Click the root product (MobilePhone) and select the **Insert->Existing Component...** command. Select the following documents and click **Open**:
   - BottomCase.CATPart
   - Battery.CATPart
   - Electronic.CATProduct
   - Body.CATPart
   - Lens.CATPart
   - Indus.CATPart
   - FrontShell.CATPart



4. Click the **Save data in Enovia V5** icon (). The Save in ENOVIA dialog box displays. Click **OK**.

5. Open the Main_Shape.CATPart file. This Part contains a document template (Keypad 1).

6. Click the **Save data in Enovia V5** icon (). The Save in ENOVIA dialog box displays. Click the  button.

7. Select the line referring to Main_Shape.CATPart and click **Modify**. Click the root product in the CATIA VPM view and click **OK**. The file is saved in ENOVIA.

8. Close all windows in CATIA.

Instantiating the Part Template

9. Load the product from ENOVIA VPM V5. To do so, proceed as follows:

- Click the Search ENOVIA icon (). The VPM Search dialog box displays.
- In the Objects scrolling list, select the **Product** option.
- If need be, enter the Owner's ID.
- Click **OK** when done. The Product ID displays in the Result window.
- Double-click the MobilePhone product.

10. Right-click the Part instances, and select the **Open...** command. The Open Modes dialog box displays. Check the **Automatic lock of all opened Parts and Documents** radio button and click **OK**. The product displays. The mobile phone support displays.
11. From the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench.
12. Click the **Instantiate from Document** icon.
13. Click the **Enovia** icon. The Search Conditions dialog box displays. Enter the name of the Revision owner and click **OK**.
14. Select the Main_Shape.CATPart in the Search Result window and click **Open**.
15. Value the **Inputs** by selecting the publications located below the Industrial Design node in the specification tree and click **OK**: The template is instantiated.

ENOVIAVPM Interoperability

Working with Assembly Templates in ENOVIAVPM

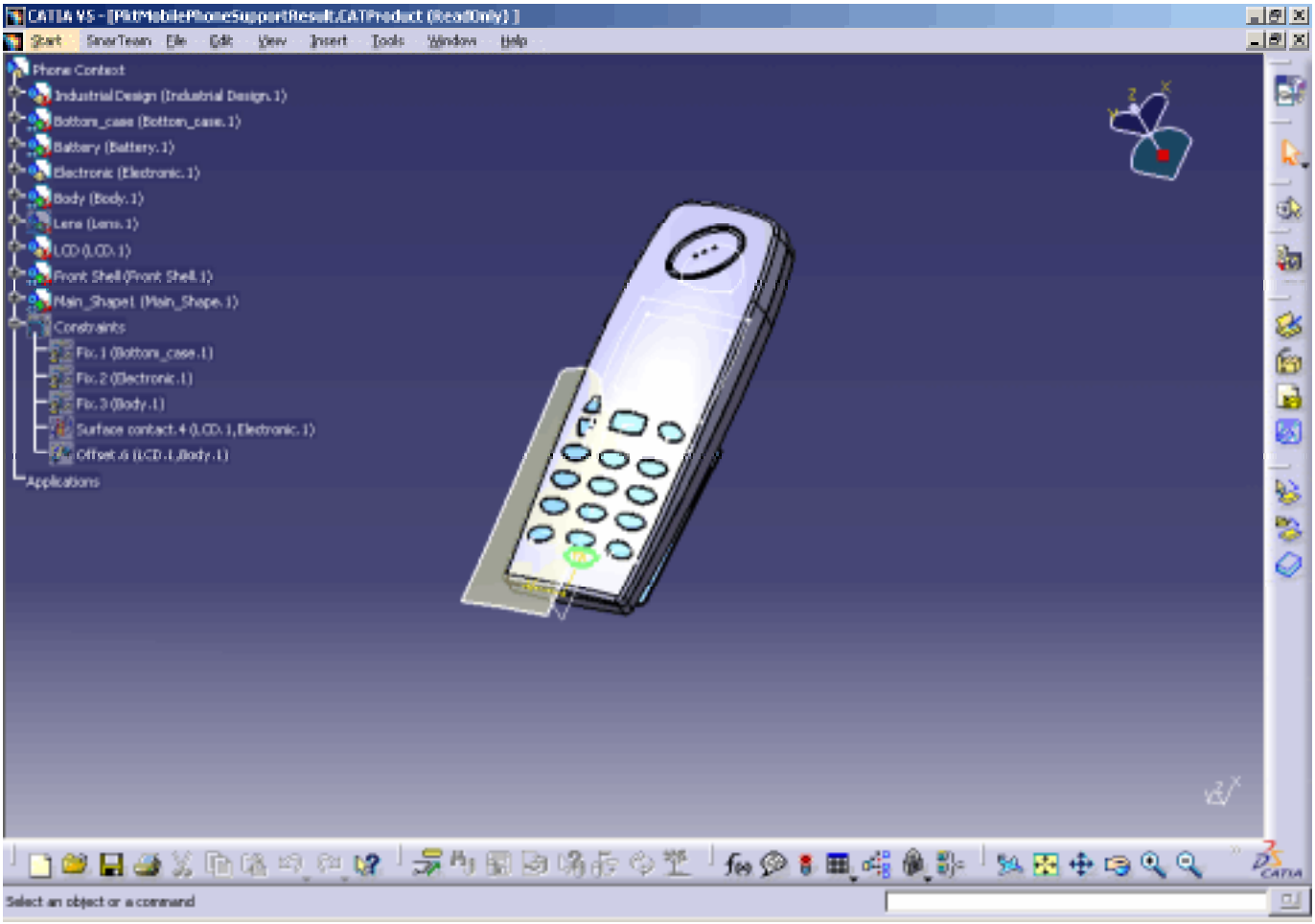
Working with Assembly Templates in ENOVIAVPM

See Working with Assembly Templates in ENOVIA.

Workbench Description

This section contains the description of the icons and menus specific to the Product Knowledge Template workbench.

The Product Knowledge Template workbench is shown below. Click the sensitive areas (toolbars) to access the related documentation.



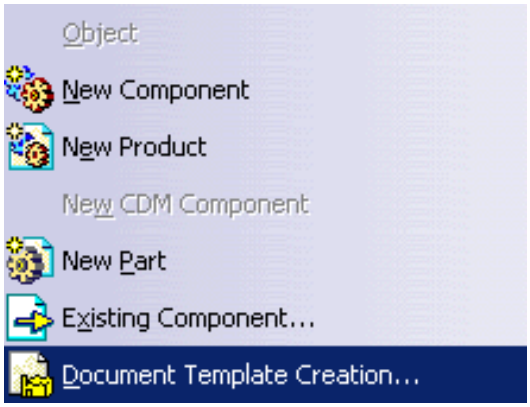
Product Knowledge Template Menu Bar
Generative Knowledge Toolbar
Templates Creation Toolbar
Templates Instantiation Toolbar

Product Knowledge Template Menu Bar

The menu specific to Product Knowledge Template is described below.

Start File Edit View Insert Tools Window Help

Insert



For	See
Document Template Creation...	Creating a Document Template

Generative Knowledge Toolbar

The Generative Knowledge toolbar contains the following tools:

See Creating a Script.

Templates Creation Toolbar

The Templates Creation toolbar contains the following tools

See Create a PowerCopy.

See Create a user feature.

See Create a document template.

Templates Instantiation Toolbar

The Templates Creation toolbar contains the following tools

See Instantiating PowerCopies, Instantiating User Features.

See Instantiating PowerCopies, Instantiating User Features.

See Instantiating PowerCopies, Instantiating User Features, and Instantiating a Part Template.

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.

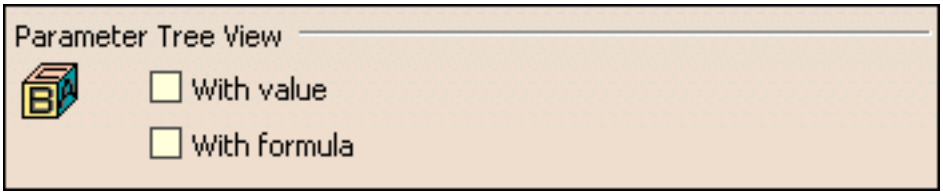
6. Click **OK** when done to validate your settings.

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.

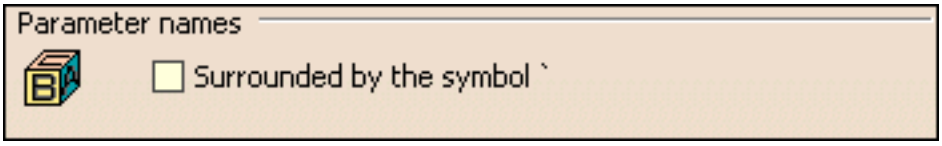
By default, this option is unchecked.

With formula

Displays the formulas constraining the parameter in the specification tree.

By default, this option is unchecked.

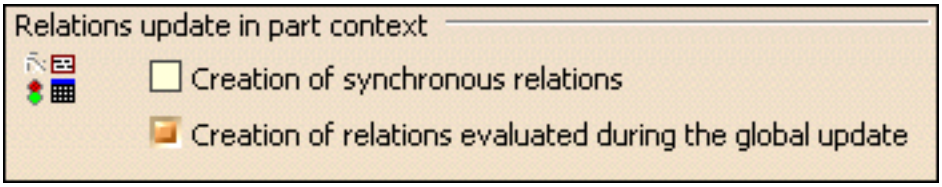
Parameter names



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:

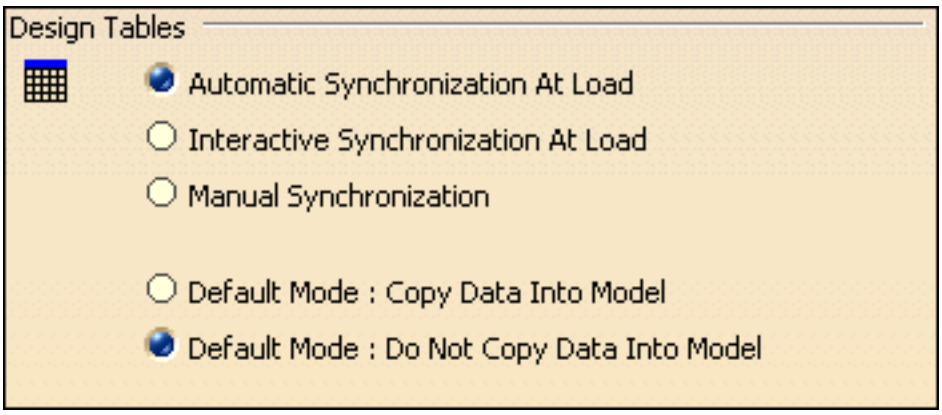
- The user wants the relations to be asynchronous
- The relation contains measures.

- 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.

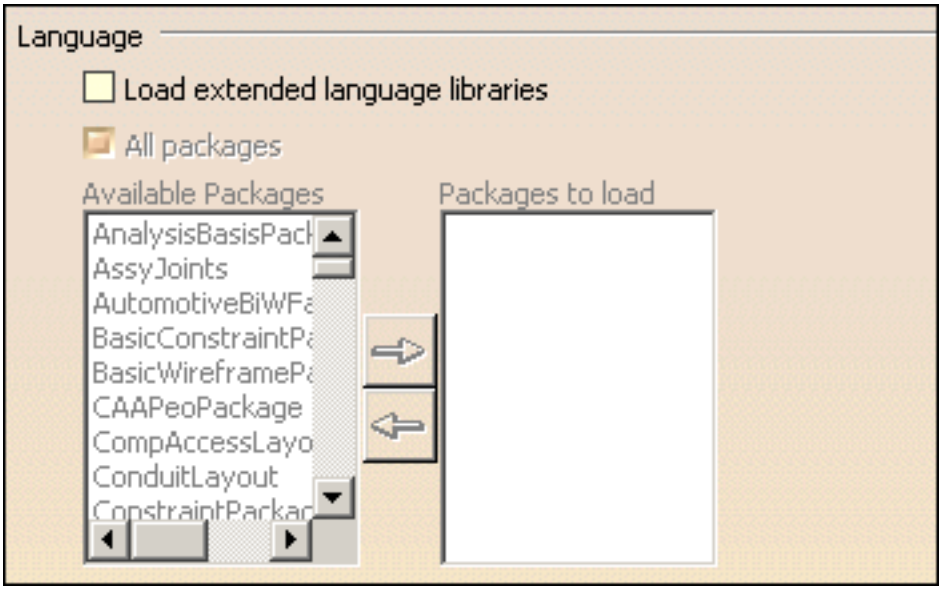
By default, this option is checked.

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.

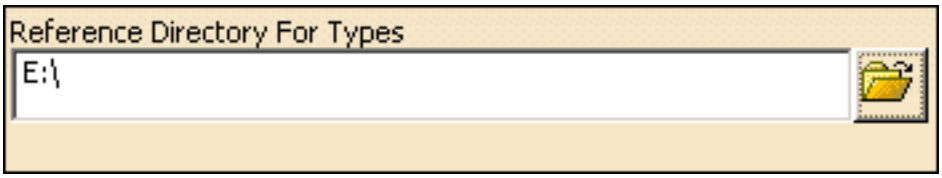
- When you open a document and some relations are broken, you might need to load all libraries to solve the error, which may take quite a long time.
- It is strongly recommended to identify the packages you will need and to select them.

By default, this option is unchecked.

All packages

Enables the user to select all packages.

Reference Directory For Types



Reference Directory For Types

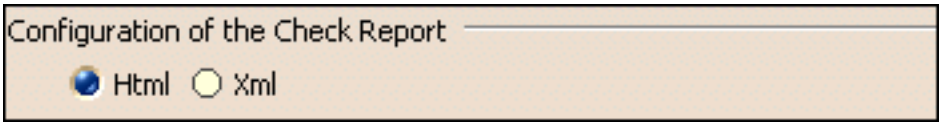
Enables the user to save the CATGScript file in the Directory indicated in the Reference Directory for Types field for later re-user (To know more, see the PKT User's Guide).

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



Html

Enables the user to generate a HTML report.

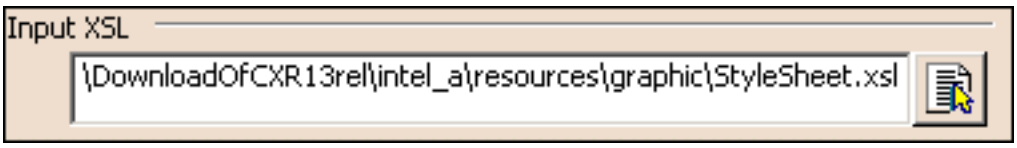
Xml

Enables the user to generate a XML report.

By default, the HTML option is enabled.

Input XSL

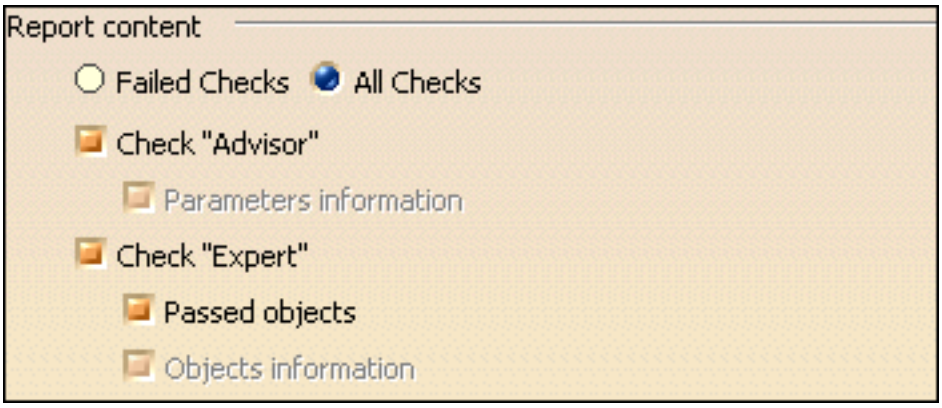
Note that this option is available only if the XML configuration setting is set.



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

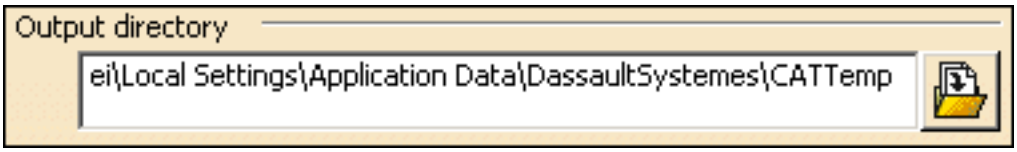
If checked, the generated report will contain information about the objects that passed the Expert checks as well as information about the parameters of these objects (diameter, depth, pitch,...).

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

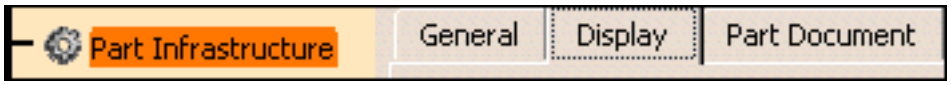
Open HTML browser into CATIA Session

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.

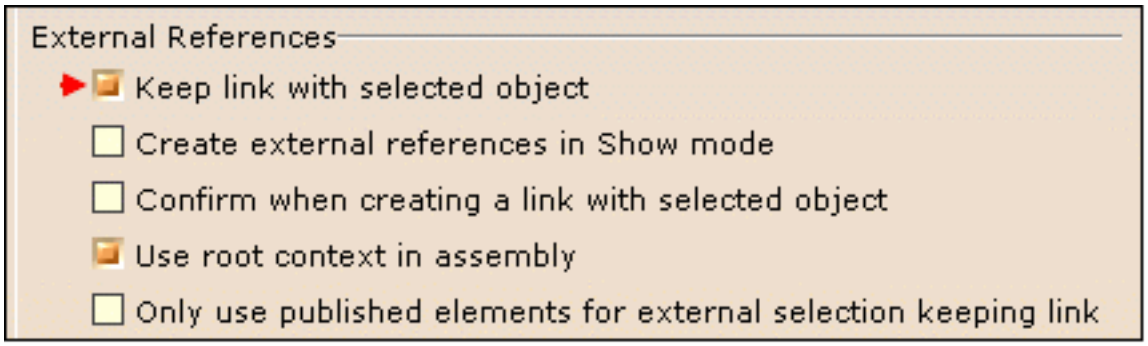
Part Infrastructure for Knowledgeware Applications



This page deals with the options concerning:

- the external references: Keep link with selected object
- the specification tree display

Part Infrastructure General option

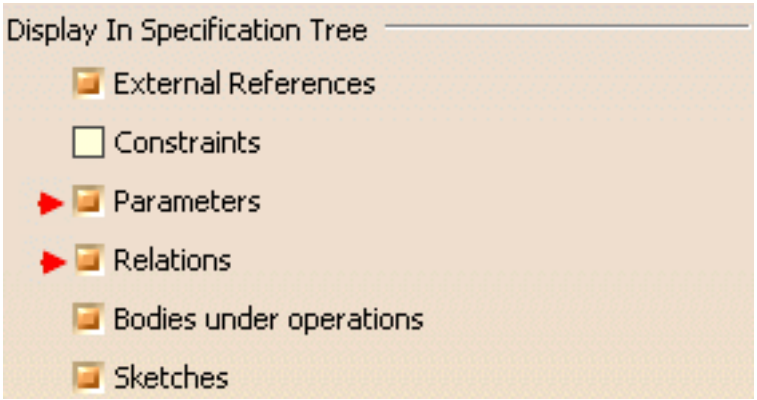


Keep link with selected object

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

C

component A component is a feature that is used to form the User Feature (or the Power Copy). This feature can be:

A geometrical feature

- A knowledge object (Rules, Formulas, Design Table,Check)
- A constraint
- Another User Feature
- An Open Body
- A Body (not the Part Body)

contextual product A product containing contextual parts whose contexts are located outside the product.

I

input An input is a feature that is not directly selected by the user to make up the User Feature. This feature points an external link through a component that is part of the UserFeature. Inputs must be valuated at instantiation.

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.

P

powercopy A set of features (geometric elements, formulas, constraints and so forth) that are grouped in order to be used in a different context, and presenting the ability to be completely redefined when pasted.

R

role The external name given to each input and published parameter. This external name is the "role". The default role of an input is the name of the feature and the default role of a parameter is its default name. For example, in the case of a parameter, it is easier to understand "The Radius of the Top Circle" rather than "Radius".
For the input, the role has another meaning: The role is useful to give more meaning to the end user, but it can also be used to automatically look for inputs. To valuate the inputs at each instantiation, you can specifically set a feature for each input, but you can also use the "Use Identical Name" function. This means that the process tries to valuate an input by looking for a feature whose name is identical to the role name of the input.

T

type A string whose recommended naming rule is the following: the first part is the company prefix and the second part is related to the current user feature.

Index

D

E

F

G

K

L

M

N

O

P

Q

R

S

T

U

V