

CATIA Infrastructure User Guide

Advanced Tasks



Site Map

Advanced Tasks

Setting Up Your Printers on UNIX and Windows

 About Setting Up Printers on UNIX and Windows

 Adding a Printer

 Removing a Printer

 Configuring an Existing Printer

 Testing the Printer

 Customizing Print Driver Plug-Ins

Using Power Input Mode

 About Power Input Mode

 Entering Data

 Running Commands

 Using the Search Language

Virtual Reality Configurations

 About Virtual Reality Support in Version 5

 Stereoscopic Viewing

- Running a Multipiped Version 5 Session
- Working With the Immersive System Assistant
- Using Head and Hand Tracking Devices in Version 5
- Using the Joystick Control Panel
- Navigating in Examine Mode
- Navigating in Fly Mode
- Installation Requirements
 - Basic Software Requirements
 - Hardware Requirements
 - How Are Version 5 Products Packaged?
 - More About the Licensing Mechanism

Using and Customizing Fonts

- About Fonts
- Customizing User Interface Fonts on Windows
- Customizing User Interface Fonts on UNIX
- Customizing Fonts for Displaying Texts
- Customizing Fonts for Displaying Geometry Area Texts
- Adding Extra PostScript Fonts
- Understanding Differences Between Font Display and Printed Output
- Recovering Custom Fonts Developed Using CATIA Version 4

Conferencing

Using Knowlegeware Capabilities

Parameters

- Introducing Parameters
- Creating a Parameter
- Copy/Pasting Parameters
- Specifying the Material Parameter
- Specifying a Parameter Value as a Measure
- Importing Parameters
- Creating Points, Lines... as Parameters
- Applying Ranges to Parameters
- Activating and Deactivating a Component
- Creating an Associative Link between Measures and Parameters
- Using Relations based on Publications at the Product Level
- Publishing Parameters
- Parameters: Useful Tips

Formulas

- Introducing Formulas
- Getting Familiar With the f(x) Dialog Box
- Using the Dictionary
 - Design Table Methods
 - Operators
 - Point Constructors
 - Evaluate Method
 - Line Constructors
 - Circle Constructors
 - List
 - Measures
 - Surface Constructors
 - Wireframe Constructors

- Part Measures
- Plane Constructors
- Analysis Operators
- Mathematical Functions

- Creating a Formula
- Creating Formulas based on Publications
- Specifying a Measure in a Formula
- Referring to External Parameters in a Formula
- Using the Equivalent Dimensions Feature
- Getting Familiar with the Equivalent Dimensions Interface
- Formulas: Useful Tips

Design Tables

- About the Design Table
- Getting Familiar with the Design Table Dialog Box
- Creating a Design Table from Current Values
- Creating a Design Table from a Pre-Existing File
- Interactively Adding a Row to a Design Table External File
- Generating a File From a Design Table
- Controlling Design Tables Synchronization
- Storing a Design Table in a PowerCopy
- Working with Design Tables in ENOVIA LCA
 - Saving a Design Table in ENOVIA LCA
 - Using a Design Table Saved in ENOVIA LCA in another Part
- Versioning a Design Table External File

- Design Tables: Useful Tips

The Knowledgeware Language

Working Through the Knowledgeware Capabilities

- Introduction - The Design Intent
- Calculating and Checking a Volume
- Working with a Design Table
- About Rule Firing
- About Automation
- Optimizing a Volume

- Appendix: Creating a Deformable Revolution Body

CATIA Knowledgeware

Integration with Enovia V5

Using the Feature Dictionary Editor

- Starting the Feature Dictionary Editor
- Creating a New Object Class
- Adding Properties to an Object Class
- Defining Discrete Values for a Property
- Generating a report
- Creating a New Feature Dictionary
- Opening a Reference Dictionary
- Modifying the Object Naming Rules

Using the Data Upward Assistant

- Using CATDUA V5 in Interactive Mode
- Using the CATDUAV5 Batch
- Viewing Results of CATDUA V5 Execution
- Return Codes Detected by the Data Upward Assistant
- List of the Detected Return Codes

Advanced Tasks

Setting Up Your Printers on UNIX and Windows

Using Power Input Mode

Virtual Reality Configurations

Using and Customizing Fonts

Conferencing

Using Knowledgeware Capabilities

Using the Feature Dictionary Editor

Using the Data Upward Assistant

Setting Up Your Printers on UNIX and Windows

About Setting Up Printers on UNIX and Windows

Adding a Printer

Removing a Printer


Configuring an Existing Printer

Testing the Printer

Customizing Print Driver Plug-Ins

About Setting Up Printers on UNIX and Windows



You can only print a document (using either the **File->Print...** command or the Quick Print icon  from the Standard toolbar) if a default printer has been set up.

On Windows, you print using the default printer declared by your Windows system administrator.

However, on UNIX, you will only be able to print once you have set up a printer.

You can set up two types of printers:

- Windows Printers
- 3DPLM Printers.

Setting up a Windows printer creates a printer configuration file, needed for printing, in `$HOME/CATSettings/Printers`.

Setting up a 3DPLM printer creates a configuration file containing the printer and the driver configuration settings in `$HOME/CATSettings/Printers/PLOTxxxx.xml`.

If you attempt to print on UNIX before setting up a printer, the following message appears:

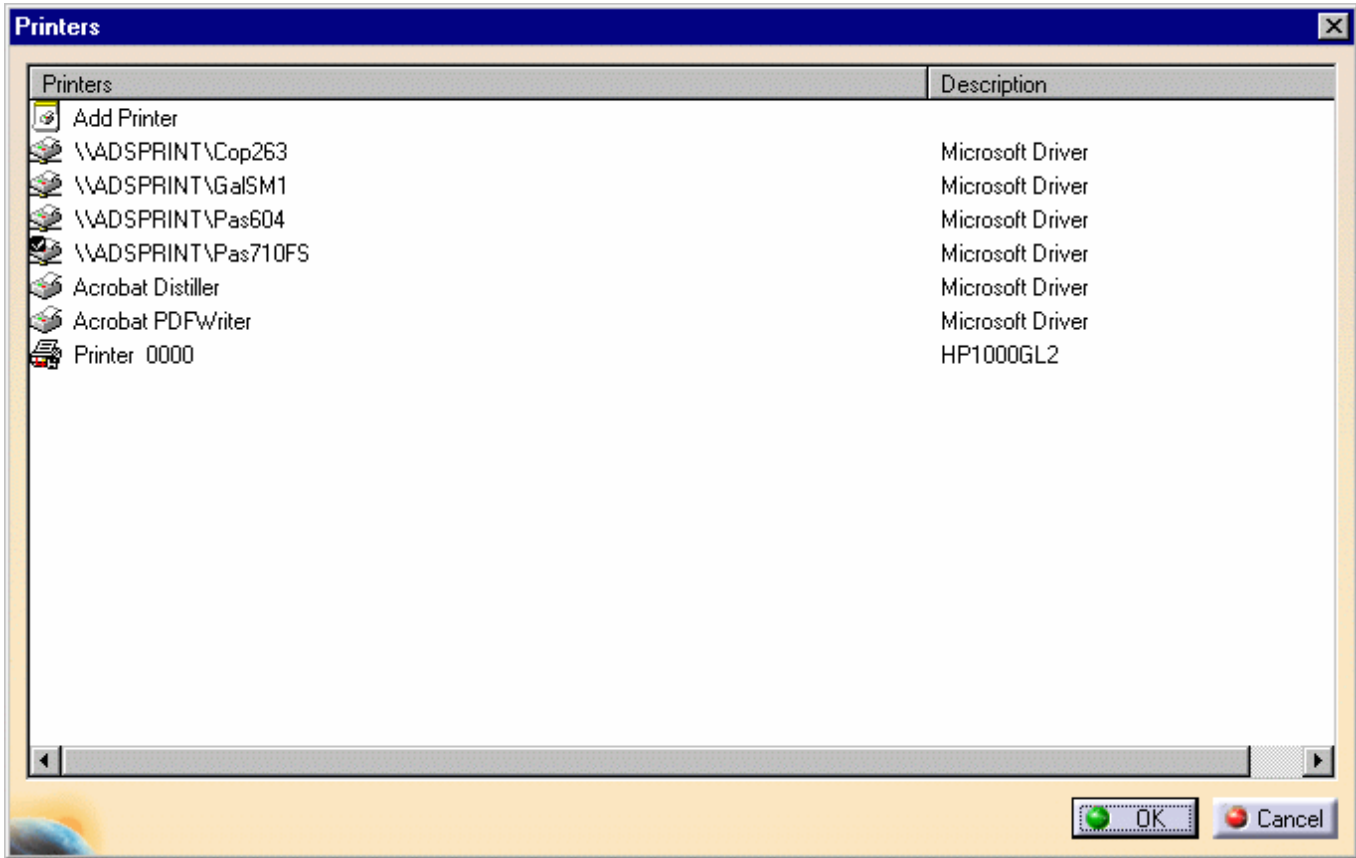


The objective of this section is to show you how to set up your environment to get your printer operational, which is performed by creating and customizing a file called a printer configuration file. The role of this file is to declare your printer so it is recognized by Version 5. Once at least one printer has been declared, the above message will no longer appear.

We assume that

- the physical printing device has been installed and connected to the system
- your system administrator has already declared your printer to your UNIX operating system.

On both UNIX and Windows, the default printer is defined using the **File-> Printer Setup...** command which opens the Printers dialog box (note that the image below shows a Windows dialog box but the look on UNIX is more or less the same):



This dialog box displays a list of available and installed printers. Note that if you are working on a UNIX workstation, you will not be able to access Windows printers.

Double-clicking one of the printers displayed in the list will open the standard setup dialog box corresponding to the selected printer type. Refer to your Windows documentation for detailed information about using this dialog box.



On Windows, you can also double-click Add Printer at the top of the list to open the Add Printer Wizard dialog box:




This dialog box enables you to choose between:

- **Windows Printer:** in that case, selecting this option then OK will open the Connect to Printer dialog box in which you will select the Windows printer you wish to add
- **3D PLM Printer:** this option lets you add printers provided by Dassault Systèmes. When clicking OK, the Printer Properties dialog box is displayed to let you define the parameters for your new printer. Refer to [Adding a Printer](#) for detailed information.

Once a printer has been added, it is displayed at the end of the list and is identified by a specific icon.

Refer to the following tasks in this section to learn more about [adding](#), [removing](#), [configuring](#) and [testing](#) a printer.

Note: if you want to set another printer as the default printer, select it from the list then right-click and choose the Set As Default contextual command. The selected printer will be set as the default printer and identified by the following icon: .



You can also set up your printers without having to run a Version 5 session. To do so:

On Windows

1. Open an MS-DOS window.
2. Change to the default folder in which you installed the product.

The default folder is:

```
C:\Program Files\Dassault Systemes\B13\intel_a\code\bin
```

3. Enter the command:

```
CATPrinterManager
```

On UNIX

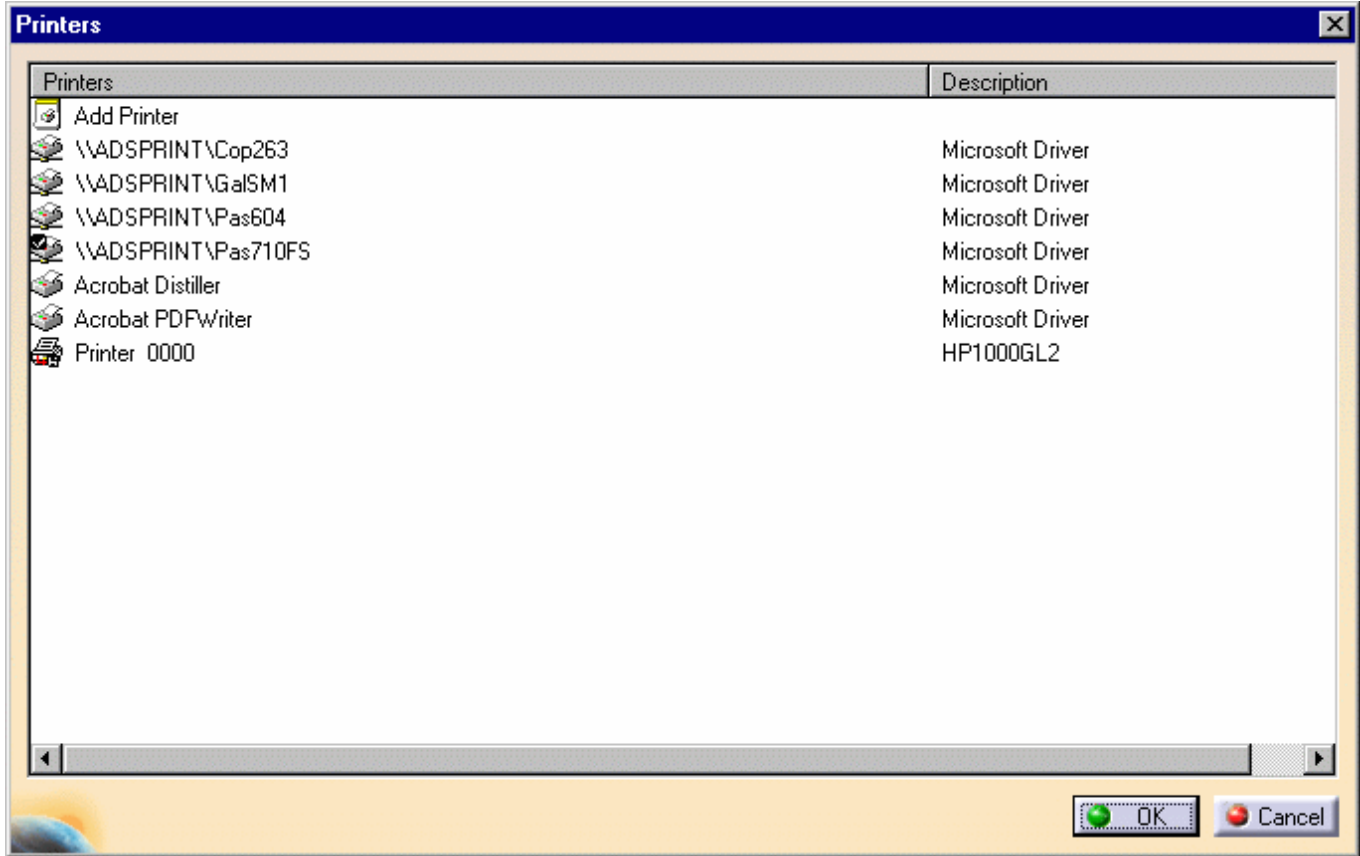
1. Change to the directory:

```
/usr/DassaultSystemes/B13/OS_a/code/command/
```

2. Run the command

```
CATPrinterManager
```

The Printers dialog box opens:



You can then set up your printers as explained above.

Adding a Printer



This task explains how to set up your printer on Windows and UNIX.



You can only print a document on UNIX if at least one printer has been set up using the **File -> Printer Setup...** command.



1. Select the **File->Printer setup** command to open the Printers dialog box then double-click Add Printer on top of the list.

If you are working on Windows, an additional dialog box named Add Printer Wizard opens:

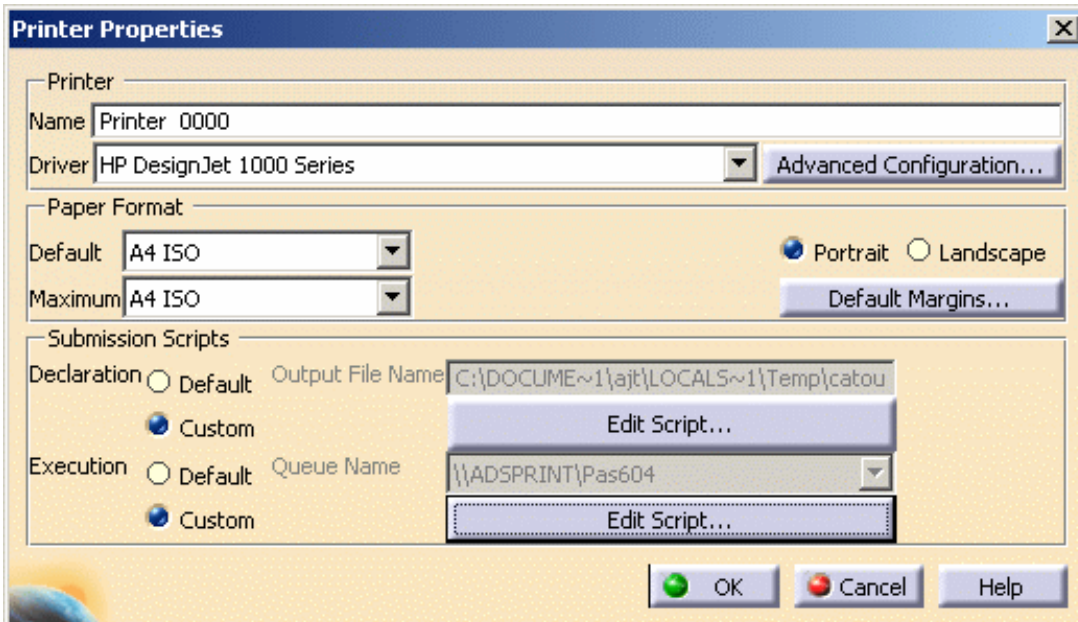


This dialog box enables you to choose between:

- **Windows Printer:** in that case, selecting this option then **OK** will open the Connect to Printer dialog box in which you will select the Windows printer you wish to add
- **3D PLM Printer:** this option lets you add printers provided by Dassault Systèmes. When clicking **OK**, the Printer Properties dialog box is displayed to let you define the parameters for your new printer.

Just click **OK** to access the Printer Properties dialog box.

If you are working on UNIX, the Printer Properties dialog box opens directly:



The above capture takes a 3D PLM Printer as an example. If you add a Windows printer, the properties dialog box will display standard setup parameters corresponding to the selected printer type. In that case, refer to your Windows documentation for detailed information about using this dialog box.

2. Click the Driver button (on UNIX) or the black arrow (on Windows) and choose the appropriate driver for the printer.

Ask your system administrator which printers require which drivers. The list of available drivers is:

- PostScript
- HP-GL/2 RTL
- HP-GL
- HP DesignJet 1000 Series
- CGM Software
- CGM
- OCE
- Raster
- V5 Print Plugin Driver
- Windows Enhanced Metafile
- PDF
- SVG.

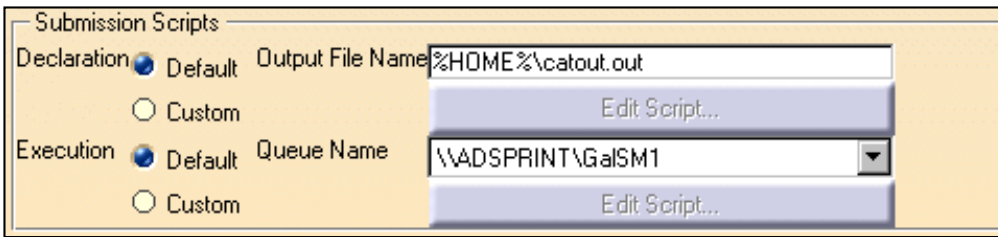
This list varies with your operating system. For instance, some of the above-mentioned drivers are not available on UNIX.

Notes:

- SVG (Scalable Vector Graphics) is a language used for describing two-dimensional graphics in XML (Extensible Markup Language) format. Three types of graphic objects can be read in this format: vector graphic shapes, images and text. SVG files can be visualized by external applications such as ADOBE products, JASC or BATIK. For detailed information, browse the following internet site:

<http://www.w3c.org>

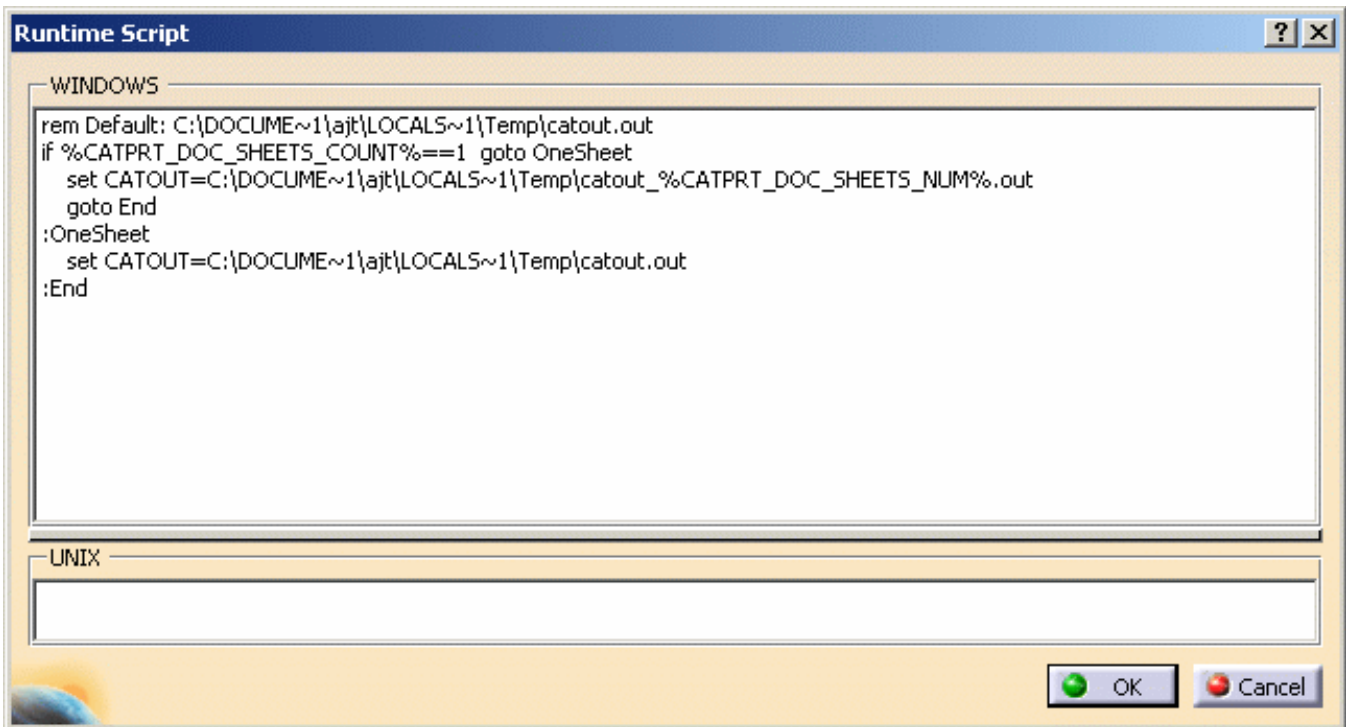
- CGM Software is intended for viewing applications but bear in mind that some restrictions apply such as: pixel images are not supported, the image size is fixed, polylines are converted into polybeziers (provided that their curves can be modified).
3. In the Paper Format group box, set the desired paper format options: default and maximum paper size, margins and orientation (portrait or landscape).
 4. In the Submission Scripts group box, set the desired options activated when the print job is submitted.



This group box lets you specify:

- the default output file name: each time you print a document, an output file is created at the location you specify in the text field
- the default queue name: this field lets you choose which print queue to send the print job to.

The Custom radio buttons let you specify the location of your own submission scripts. Clicking these buttons activates the **Edit Script...** buttons to let you modify the script in the Runtime Script window:



The **Advanced Configuration...** button lets you define advanced configuration settings for the driver. Note that once a printer has been added to the list of printers, you can also access advanced configuration settings by right-clicking the printer in the list then selecting the **Configure** contextual command.

Refer to [Configuring an Existing Printer](#) for detailed information.

5. Click **OK** to return to the Printer Setup dialog box.

A printer configuration file containing all the settings you set in the Printer Properties dialog box is created in: \$HOME/CATSettings/Printers/PLOT0000.xml which is the default location. The numbers of additional printers created are incremented by one, as follows: PLOT0001.xml, PLOT0002.xml, etc. However, note that when printing to 3DPLM printers, you can now modify the default location of the configuration file via the [Printers](#) tab.



The format of Version 4 plot configuration files is not compatible with the Version 5 format.



Removing a Printer



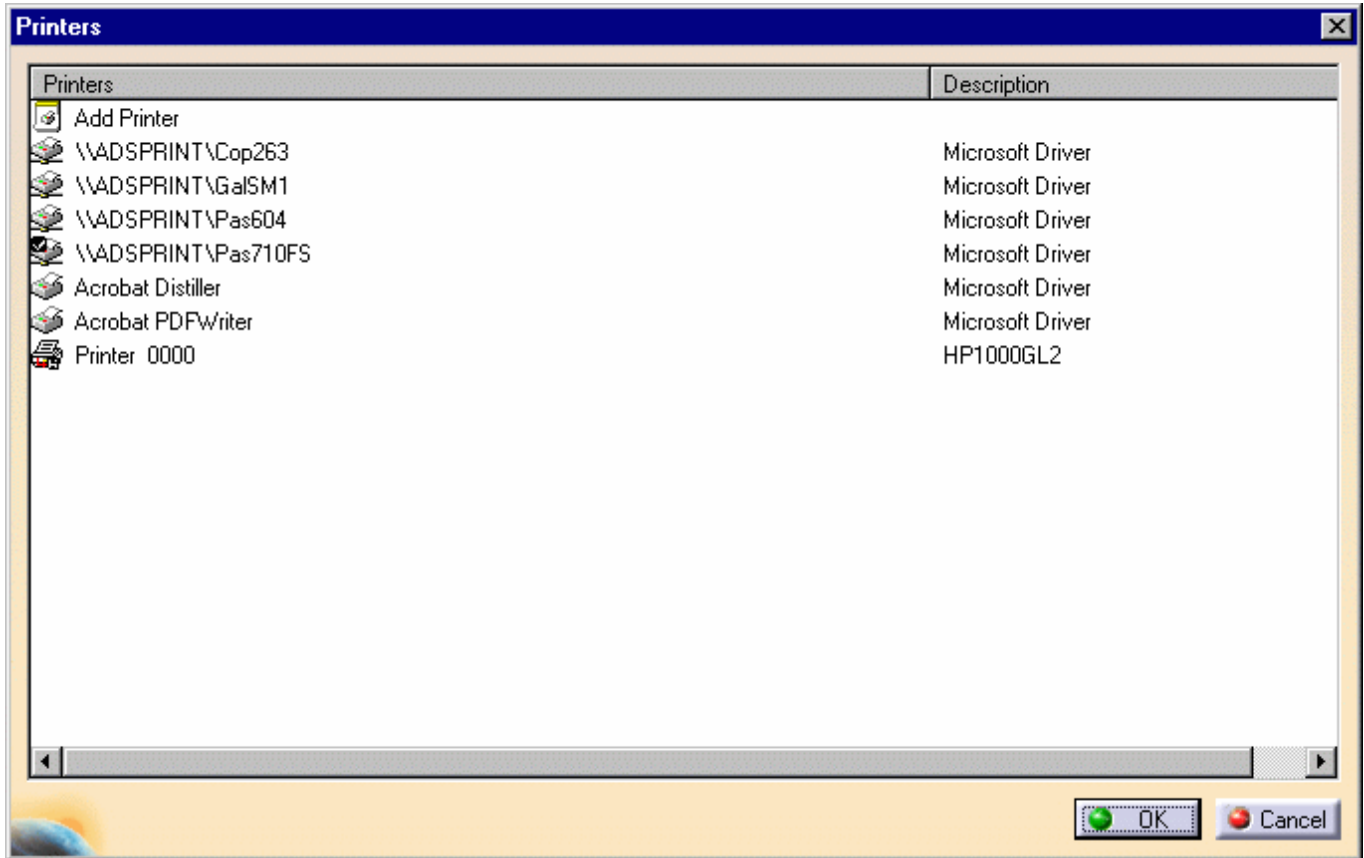
This task explains how to delete a printer.



At least one printer must have been set up using the **File -> Printer Setup...** command.



1. Select the **File->Printer setup** command to open the Printers dialog box:



2. Select the printer to be deleted from the list of printers

3. Right-click then select the Remove command to remove the printer.

The printer configuration file located in \$HOME/CATSettings/Printers/PLOTxxxx.xml (the default location) is deleted. However, note that when printing to 3DPLM printers, you can modify the default location for the configuration file via the [Printers](#) tab.

Note: you cannot remove Windows printers.

Configuring an Existing Printer



This task explains how to reconfigure an existing printer.

About Configuration Files for 3DPLM Printers

Up to V5R11

In previous releases, the driver and printer configuration files were stored apart in permanent settings. Permanent setting files store customization you perform mainly using the various tabs provided by the **Tools->Options...** command.

Permanent print settings were created by default in a location referenced in the Version 5 runtime environment by the `CATUserSettingPath` variable:

- in "C:\Documents and Settings\user\Application Data\DassaultSystemes\CATSettings" (referred to as "\$HOME" in this scenario) on Windows
- in the CATSettings directory in your home directory (also referred to as "\$HOME") on UNIX.

The driver and printer configuration files were stored in two different files defined as text files:

- the printer configuration file was stored in `$HOME/CATSettings/Printers/PLOT0000.plot_cfg`
- the driver configuration file was stored in `$HOME\CATSettings\Print.CATSettings` and could not be modified.

From V5R12 onwards

For the sake of convenience, a single configuration file in XML format is now used to configure the printer and the driver. By default, the configuration file will still be stored in the permanent setting files created in a location referenced by the `CATUserSettingPath` variable. This location is:

- `$HOME/CATSettings/Printers/PLOT0000.xml` (on Windows)
- `$HOME\CATSettings\Printers\PLOTxxxx.xml` (on UNIX).

However, you will see further in this guide that you can choose a directory other than the default one proposed.

Any modification entered in the Printer Properties dialog box (detailed below) will be written in XML language in the configuration file. There are as many .xml files as there are 3DPLM Printers and the numbers of additional printers are incremented by one as follows: `PLOT0000.xml`, `PLOT0001.xml`, `PLOT0002.xml`, etc..

Note that configurations from previous releases are also supported, which means that, if your printer configuration file is in `.plot_cfg` format, you will be able to translate it into XML format.

About Configuration Files for Windows Printers

As far as Windows printers are concerned, the application uses the driver and printer setting files installed on your computer.

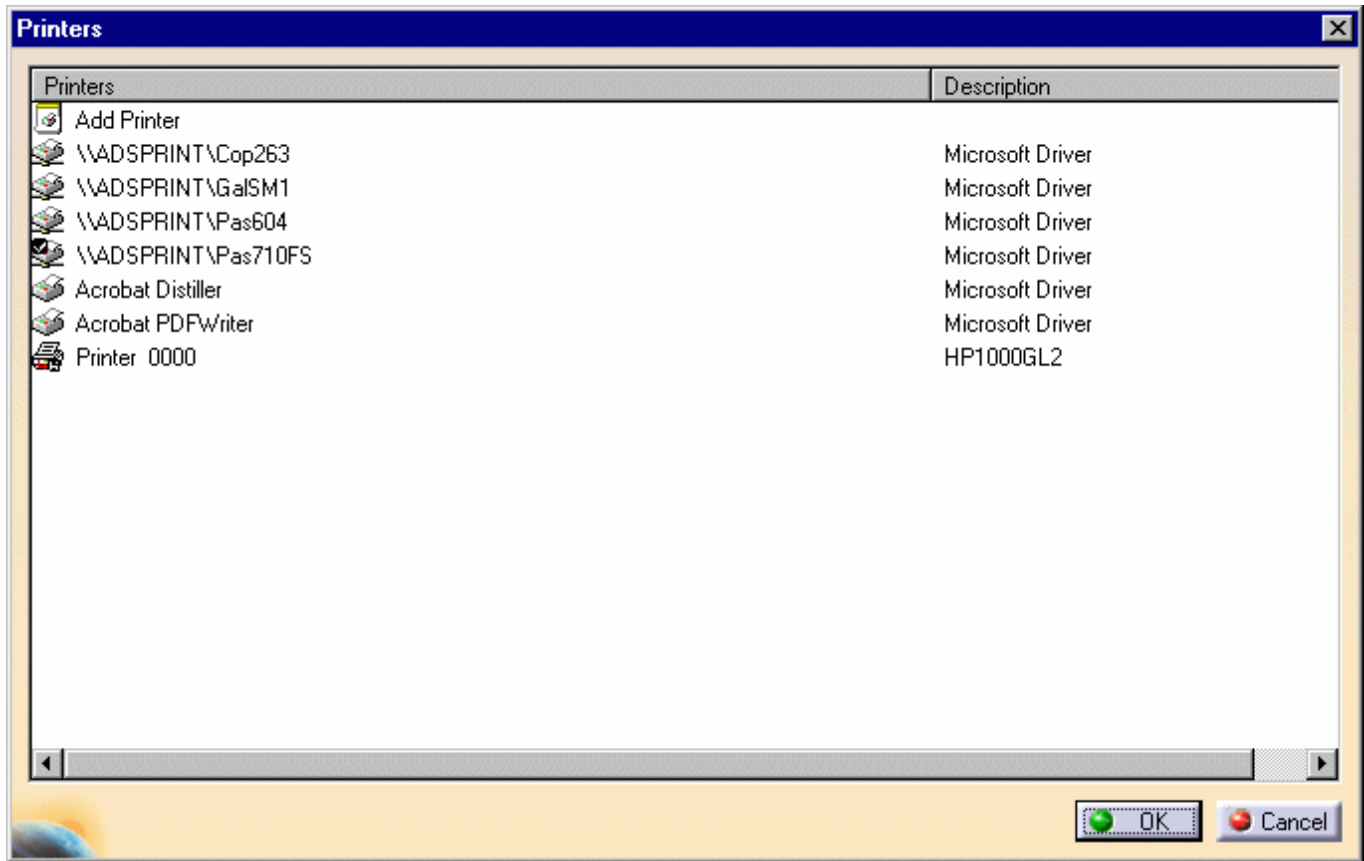
Configuration Example



At least one printer must have been set up using the **File -> Printer Setup...** command.

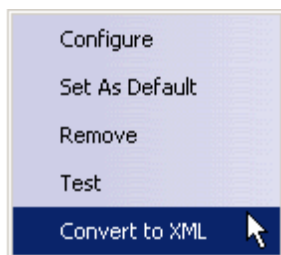


1. Select the **File->Printer setup** command to open the Printers dialog box:



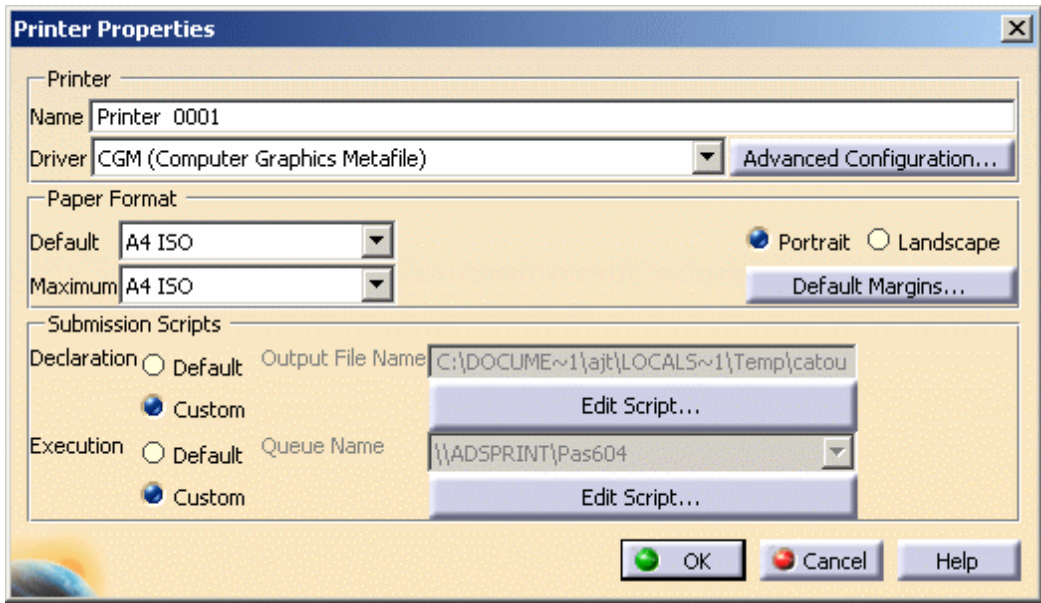
2. Select the printer to be reconfigured from the list of printers.

3. If the printer you are configuring is associated to a `.plot_cfg` file, right-click the printer from the Printers dialog box then select the **Convert to XML** contextual command:



A configuration file in XML format replaces the `.plot_cfg` configuration file.

4. Right-click the printer from the list then select the **Configure** contextual command (or double-click the desired printer name) to access the Printer Properties dialog box.



The above capture takes a 3D PLM Printer as an example. If you add a Windows printer, the properties dialog box will display standard setup parameters corresponding to the selected printer type. In that case, refer to your Windows documentation for detailed information about using this dialog box.

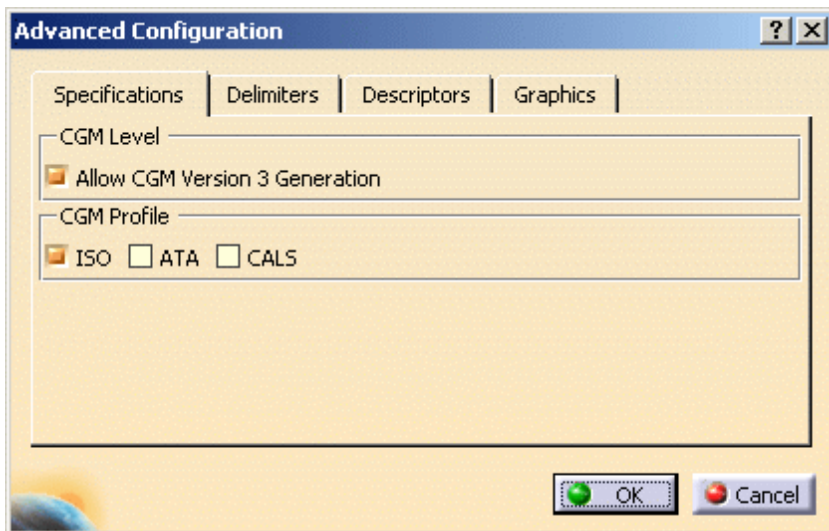
5. Modify the necessary data in the properties dialog box.

Refer to [Adding a Printer](#) for detailed information on the printer properties.

6. Click the **Advanced Configuration...** button to configure the driver.

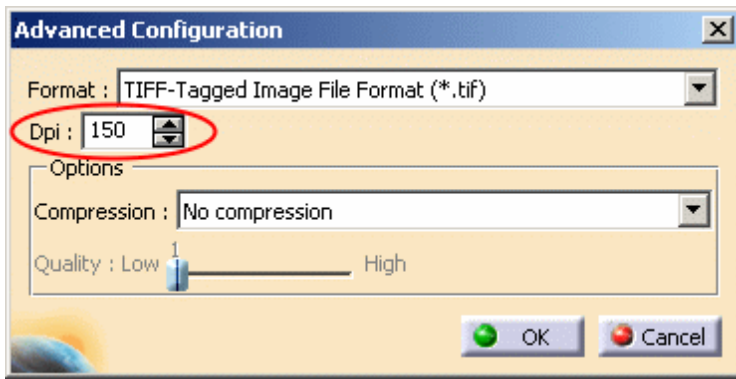
Driver configuration options vary from one driver to another. Refer to your plotter documentation for information about the configuration options for each driver.

Note that you cannot access advanced configuration options for SVG drivers.



7. Access the desired tabs then make the necessary modifications.

Note: As far as Raster drivers are concerned, you can now set more precisely the desired DPI value using the Dpi spin box:



8. Click **OK** to validate and close the Advanced Configuration dialog box then click **OK** again to close the Printer Properties dialog box.

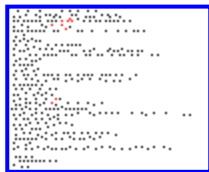
The printer configuration file (\$HOME/CATSettings/Printers/PLOTxxxx.xml) is modified.



When printing to a 3DPLM printer, the default location for the configuration file (containing the configuration settings for the driver and the printer) can be modified if needed in the [Printer Creation Directory](#) area under the Printers tab.

9. Access the printer configuration file then open it, you will see that your modifications have been written in XML format in the file.

In our example, we have changed the paper format from "A4 ISO" to "B ANSI" and the CGM Profile from "ISO" to "ATA". These modifications have been highlighted in red in the sample file below for greater clarity. Just click the thumbnail to see the full-size picture:



Note that a .dtd file containing the description of the XML configuration file is provided in:

- C:\Program Files\Dassault Systemes\Bn\intel_a\resources\printerDTD\printer.dtd (on Windows)
- /usr/DassaultSystemes/Bn/OS_a/resources/printerDTD/printer.dtd (on UNIX)

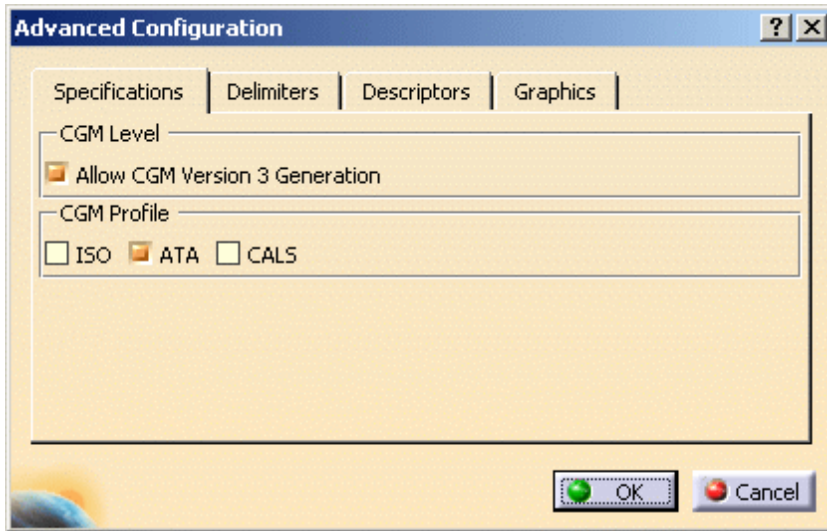
where "n" is the current release number

and

"OS_a" is:

- aix_a
- hpux_b
- irix_a
- solaris_a.

10. Re-select the advanced configuration settings to check that your modifications have been taken into account:



As shown in the above screen grab, your settings have been saved and the CGM Profile is "ATA" and "ISO".

Testing the Printer



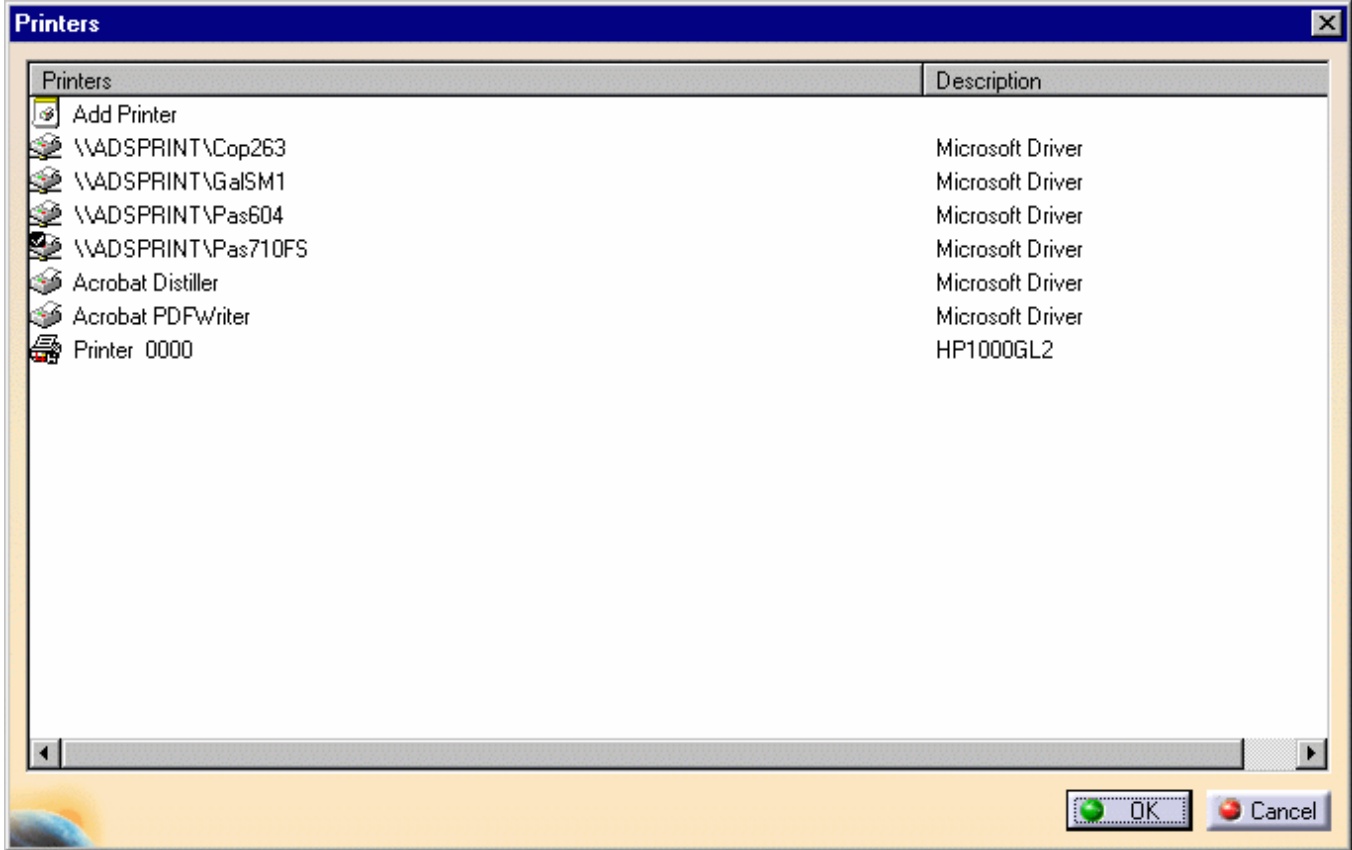
This task explains how to test your printer on UNIX.



You need to have already set up at least one printer.



1. Select the **File->Printer setup** command to open the Printers dialog box:



2. Select the printer to be tested.

3. Right-click then select the **Test** command.



Customizing Print Driver Plug-Ins



You can use print drivers other than those provided with Version 5. This task provides a step-by-step scenario describing how to plug-in your own print drivers in your Version 5 environment without CAA context using CATPrDrvPlugins.

A CATPrDrvPlugin is made of one group of 10 functions and one shared data structure:

- **Plug-in methods** are functions you implement in the plug-in. These functions will then be called by Version 5. Their names all begin with "CATPDP_", for instance CATPDP_Begin
- **Data structures** are plug-in-specific types defined for use in the plug-in API. Their names begin with "CATPDP", for instance CATPDPPParameters. All plug-in API structures and definitions are found in CATPDPluginAPI.h located in `SCATStartupPath\startup\PrintServices\PrintDriverPlugIn`.

Click the thumbnail below to find a description of the structure of CATPDPPParameters:



A plug-in is a native code library whose source conforms to standard C syntax.

The plug-in file type depends on the platform you are working on:

- Windows: .DLL (Dynamic Link Library) files
- UNIX: .SO or .DSO (Shared Objects) files.

You can use the template file (CATPluginTemplate.cpp) located in `SCATStartupPath\startup\PrintServices\PrintDriverPlugIn` and adapt it to your needs by implementing the plug-in methods provided by Version 5.

When the code has been written, you will have to compile and build the module then declare the resulting module to the Print Driver Manager File. The file CATPDPlugin.mak provided in `SCATStartupPath\startup\PrintServices\PrintDriverPlugIn\BUILD\WINDOWS` can help you build your library module.



1. Access the file CATPluginTemplate.cpp located in

`SCATStartupPath\startup\PrintServices\PrintDriverPlugIn`

2. Write the code corresponding to the interfaces to implement the plug-in methods that will be called by Version 5.

For information about the syntax and use of C or C++ language, refer to your language documentation.

The provided methods are listed in the table below. Click the desired method to access the corresponding description:

Method Name	Description
CATPDPP_Begin	Driver initialization with parameters provided by user interface
CATPDP_End	Indicates if the plug-in driver is about to be closed or removed
CATPDP_DefineColor	Defines a RGB color in the palette
CATPDP_SelectDrawColor	Selects the drawing color
CATPDP_SetDrawWidth	Selects the drawing path
CATPDP_MoveTo	Moves the pen without drawing
CATPDP_LineTo	Draws a line
CATPDP_SelectFillColor	Selects the filling color for polygon primitives
CATPDP_FillArea	Fills a polygon
CATPDP_DrawBitmap	Draws a bitmap

CATPDP_Begin

Provide global initialization for the plug-in driver.

Syntax

```
#include < CATPDPluginAPI.h >
CATPDPErr CATPDP_Begin(const CATPDPPParameters& iParam);
```

Parameters

iParams Global structure data parameters.
 Various user printing options

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

This is the first call used from the V5 Printer Manager to inform the driver plug-in the the printing is started. The Printer Manager fills the CATPDPPParameters according to the user choices. This structure is then passed to the plug-in by calling the CATPDP_Begin function.

[Back to Method summary](#)

CATPDP_End

Provide global deinitialization for the plug-in driver.

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErr CATPDP_End(void);
```

Parameters

No parameters

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

The V5 Printer manager calls this function once the printing is done. Be sure to release allocations , instances or file I/O flushing in this function. After this call the plug-in driver is unloaded and the printing stopped.

[Back to Method summary](#)

CATPDP_DefineColor

Defines a color in the palette

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErr CATPDP_DefineColor(int iIndex , float iRed, float iGreen, float iBlue);
```

Parameters

iIndex Index in the color table (between 0 and 255)
iRed Red color in RGB coordinates (between 0 and 1)
iGreen Green color in RGB coordinates (between 0 and 1)
iRed Blue color in RGB coordinates (between 0 and 1)

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Defines a color in rgb coordinates. This index is used for function [CATPDP_SelectDrawColor](#) or [CATPDP_SelectFillColor](#).

[Back to Method summary](#)

CATPDP_SelectDrawColor

Selects the drawing color

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErr CATPDP_SelectDrawColor(int index);
```

Parameters

iIndex Index of the current drawing color (between 0 and 255)
used for next drawing primitive

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Selects the current drawing color related to the index defined by the CATPDP_DefineColor.

[Back to Method summary](#)

CATPDP_SetDrawWidth

Selects the current draw width

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErr CATPDP_SetDrawWidth (float iWidth);
```

Parameters

IWith Thickness in mm of the current drawing with
used for next drawing primitives

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Selects the current draw width. The width is in mm.

[Back to Method summary](#)

CATPDP_MoveTo

Moves the pen without drawing

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErrror CATPDP_MoveTo(float iX, float iY);
```

Parameters

iX X coordinate to move
iY Y coordinate to move

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Moves the pen to (x, y) in device coordinates without drawing. Coordinates are in mm.

[Back to Method summary](#)

CATPDP_LineTo

Draws a line

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErrror CATPDP_LineTo(float iX, float iY)
```

Parameters

iX X coordinate to draw
iY Y coordinate to draw

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Draws a line from the current pen position to (x, y) in device coordinates with current draw color, line type and draw width.

[Back to Method summary](#)

CATPDP_SelectFillColor

Selects the filling color

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErrors CATPDP_SelectFillColor (int iIndex )
```

Parameters

iIndex Index of the current Fill color (between 0 and 255)
 used for next Fill primitive

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Selects the current filling color related to the index defined by the CATPDP_DefineColor.

[Back to Method summary](#)

CATPDP_FillArea

Fills a polypolygon

Syntax

```
#include < CATPDPluginAPI.h >  
CATPDPErrors CATPDP_FillArea(int iOutlines, const int* iCorners, const float* iCoord)
```

Parameters

iOutlines number of polygons in the polypolygon.
iCorners Array of outlines integers giving the number of corners in each polygon.
iCoord array of floats giving the coordinates of all corners of each polygon

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Fills a polypolygon in device coordinates with current fill color.

[Back to Method summary](#)

CATPDP_DrawBitmap

Draws a bitmap

Syntax

```
#include < CATPDPluginAPI.h >
```

```
CATPDPErr CATPDP_DrawBitmap(float iX, float iY, const int iTypeOfBitmap, const unsigned char* iPixels, const int iSize);
```

Parameters

iX	X coordinate of the bitmaps to draw
iY	Y coordinate of the bitmaps to draw
iTypeOfBitmap	color mode of bitmap: 0 = RGB , 1 = Black and White bitmap
iPixels	array of pixels
iSize	size in bytes of the pixel array

Returns

- If successful, the function returns CATPDP_NO_ERROR.
- If unsuccessful, the plug-in the function returns an error code.

Description

Draws a bitmap in device coordinates at the given (x, y) position.

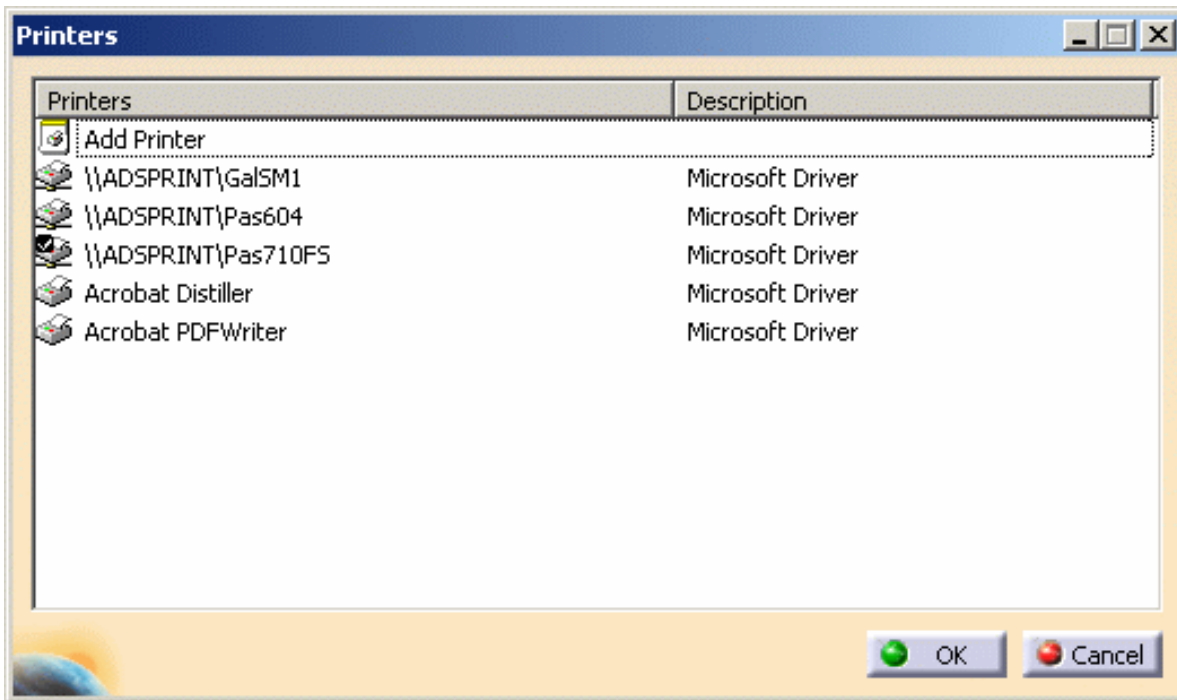
[Back to Method summary](#)

3. Compile and build the module.

Note: you can use the file CATPDPlugin.mak provided in

`%CATStartupPath\startUp\PrintServices\PrintDriverPlugIn\BUILD\WINDOWS.`

4. In your Version 5 session, select the **File->Printer Setup...** command to open the Printers dialog box:



5. Double-click Add Printer.

- If you are working on Windows, an additional dialog box named Add Printer Wizard opens:

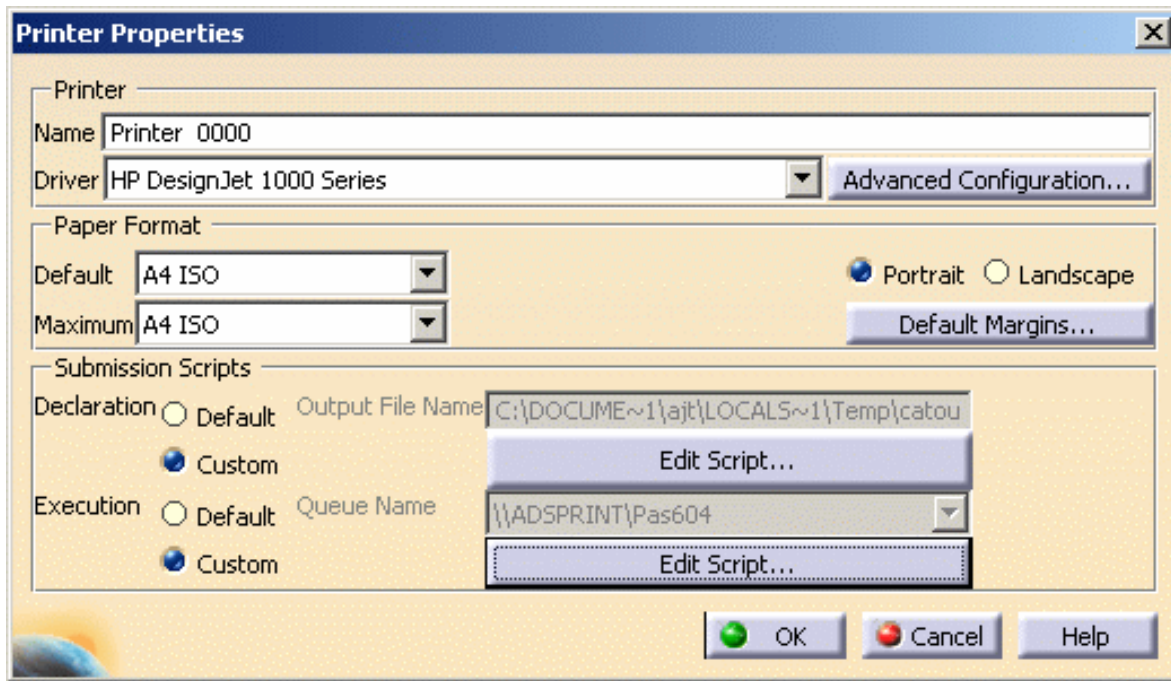


This dialog box enables you to choose between:

- Windows Printer: in that case, selecting this option then OK will open the Connect to Printer dialog box in which you will select the Windows printer you wish to add
- 3D PLM Printer: this option lets you add printers provided by Dassault Systèmes. When clicking OK, the Printer Properties dialog box is displayed to let you define the parameters for your new printer.

Just click OK to access the Printer Properties dialog box.

- If you are working on UNIX, the Printer Properties dialog box opens directly:



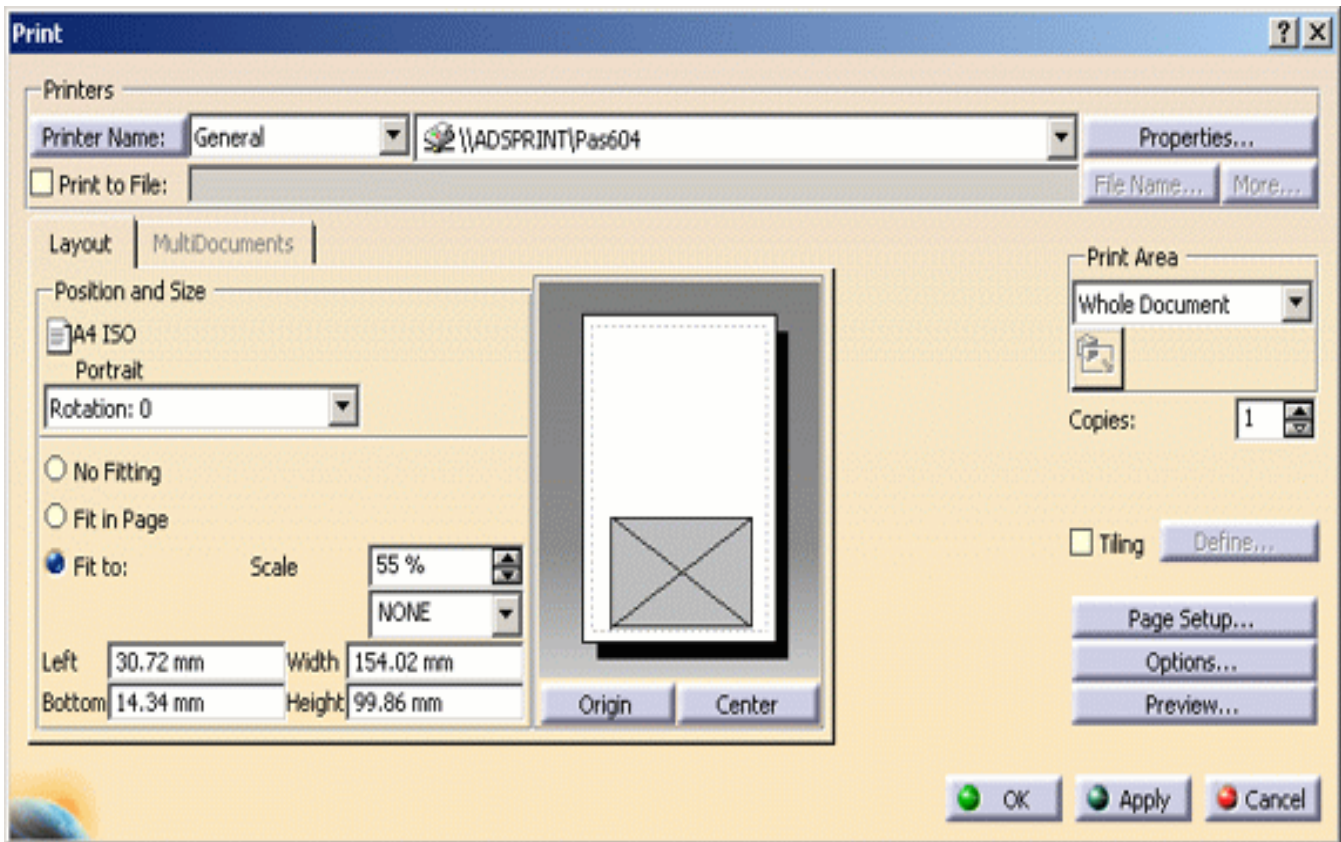
6. Select **V5 Print Plugin Driver** from the Driver pulldown list.

7. Click the **Advanced Configuration...** button to access the following dialog box:



8. In the Plugin Driver selection field, enter the path of the library module you built in step 3 or click the **Plugin...** button which enables you to browse your file tree.

9. Select the **File->Print...** command to open the Print dialog box:



10. Indicate the name of the new custom printer in the Printer Name field.

11. Modify the desired print settings (refer to [Customizing Print Settings Before Printing Your Documents](#) for detailed information) then click OK to confirm and print your document.




Using Power Input Mode



- About Power Input Mode
- Entering Data
- Running Commands
- Using the Search Language

About Power Input Mode

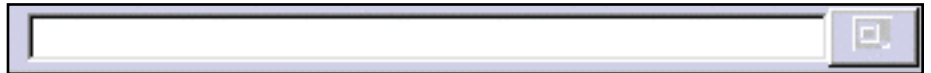


-  Power input mode is a user-friendly productivity assistant allowing you to:
- enter numeric data more easily in editable fields and spinners (but cannot be used in combo lists); as such, it is intended as an alternative to typing values in dialog boxes (dialog boxes are however still available)
 - enter commands directly (by typing the c: followed by the name of the command)
 - entering selection queries using the query language available for the **Edit->Search...** command.

Power input is available in certain (but not all) application commands.

To activate power input mode, select the **Tools->Options...** command, click the General tab, and click the CATIA - P2 option, then restart your session.

The power input field is located in the bottom right corner of the status bar:



Entering Data



This task shows you how to enter data more rapidly and more productively.




1. Select a command that allows you to use the power input field for data entry.

For example, in the Part Design application, you could select the Chamfer command.

In the lower right corner of the window, the power input field is displayed:



If we zoom on the lower right corner of the window, note that:

- the area to the left of the power input field displays the names of the dialog box options for which power input is possible: this is the case only for editable fields and spinners (but not for combo lists)
- the power input field contains the default values for those options, separated by commas
- the  icon appears to the right of the power input field: when this icon is displayed, you know that the command you are using supports power input.

The dialog box remains open by default.

2. Enter the values for all the options in the power input field, making sure that you separate each entry using the separator appropriate for your language environment, then press ENTER to validate your input.

Even if the cursor focus is in a document window, you do not need to click in the power input field to transfer the focus to the power input field:


- any characters you type will be directed to the power input field automatically
- use the Tab key to scroll from the power input field in the main application window to the data input fields in application dialog boxes
- select the ESCAPE key to return the focus to the document window.

This mechanism is also implemented in drafting documents when you enter text: as soon as you start typing, the focus is transferred automatically to the text editor window.

On Windows, you can set the separator you want to use. To do so, select the **Start->Settings->Control Panel** command, double-click the Regional Settings control, then click the Number tab. Set the separator using the List Separator option. The default is the comma (",") for the English, Japanese, Korean and simplified Chinese environments, and the semi-colon (";") for all other supported environments.

On UNIX, you cannot set the separator. The separator is the comma (",") for the English, Japanese, Korean and simplified Chinese environments, and the semi-colon (";") for all other supported environments.

The values you enter in the power input field are updated instantaneously in the fields in the dialog box.

If the dialog box is still open, click **Apply** or **OK** in the dialog box. However, you can also remove the dialog box by clicking the  icon. If the dialog box is no longer visible, pressing ENTER validates your input and executes the command (equivalent to **Apply** and **OK**).

If the color of the text you enter in the power input field changes to red, this means that you have made a mistake: for example, the number of values you enter may exceed the number of options for which power input is possible.

Use the up and down keyboard arrow keys if you want to recover any input you previously entered.

Finally, if a dialog box provides contextual commands, right-clicking over the power input field also accesses the same commands.

Using power input helps you to be more productive because it provides an alternative to using dialog boxes for inputting data. In dialog boxes, you have to click in each editable field, or use the Tab key, to move from field to field. The power input field lets you concentrate on just the data you have you enter, and thereby facilitates data input.



Running Commands



This task explains how to run commands from the power input field.



1. To run a command from the power input field, enter a command like this:

c: command_name

where *command_name* is the name of the command as it appears in the menus.

For example, enter the following command:

c:New...

or:


c:New

to run the **File->New...** command.

2. Press the ENTER key to run the command.



When pointing at icons, the syntax for running the associated command is displayed to the left of the power input field, to remind you that you can also run the command from

the power input field. For example, when you point at the New  icon, the following message is displayed:

c:New



Using the Search Language



This section explains how to use the search language to search for objects.

The search language offers almost all of the search functions available with the **Edit->Search...** command described in "[Selecting Using the Search... Command](#)". Searching using this command generates a search query (expressed in the search language) displayed in the "Generated query" field of the Search dialog box.

The search language can now be used both in the power input field and in the Advanced tab of the Search dialog box. The search query both searches for the elements and automatically selects them.

Note also that any generated query can be run via the power input field, whether *transformat* or not.



1. Enter the search string.
2. Press the ENTER key.

The searched objects are sent to the current command. If the current command is the **Select** command, the objects are selected.

Search Language: Syntax

You can search for objects using the same criteria as with the **Edit->Search...** command.

The message catalog `KeyboardInput.CATNls` sets up the power input search syntax, and search language shortcuts. The localized version of this message catalog determines the exact syntax and shortcuts for each language.

Operating Signs

The search language uses the following separators (whose role you will discover in the examples below):

- `:` and `=` (these separators are interchangeable)
- `!=` (different)
- `<`, `<=`, `>`, `=>`

Searching by Name

You can search for an object name displayed in the specification tree. This is particularly useful if you renamed objects using the **Edit->Properties** command, or the **Properties** contextual command. The name can also contain special characters.

To search for an object by its name, enter the following command:

name:object_name or *name=object_name*

or a command using an abbreviation referred to as a "shortcut" as follows:

n:object_name

where "object_name" is the name of the object.

You can also use the "*" character as a wildcard to replace any number of characters. For example, the command:

name:wheel*

searches for all objects starting with the string "wheel".

The message catalog KeyboardInput.CATNls sets up unambiguous default shortcuts. For example, there is no ambiguity between the shortcut "c:" (used in the power input area to enter a command), and "col:" (used for searches on color). In localized versions of the catalog, check that there are no identical shortcuts for two different items.

Searching by Name in Graph

You can search for an object name as it is displayed in the specification tree. This search is different from searching by name since it deals with the name as it is displayed in the specification tree.

This name may be customized and thus, may differ from the name you entered in the Properties dialog box (for instance, if you set a small fixed size for tree items in the Tree settings then the name appears truncated in the tree).

To search for an object by its name as displayed in the specification tree, enter the following command:

name in graph= *wheel*

searches for all objects containing the string "wheel".

If you want the search to be case sensitive, enter the following command:

NAME IN GRAPH= *Wheel*

Searching by Type

Use the Type field in the Search dialog box to display a list of types (the types are translated in each language).

To search for an object by its type, enter the following command:

type: type or *type=type*

or:

t: type

You can also search for types using the "." (period) as follows:

For example:

'Part Design'.Pad

searches for all objects of type Pad created using the "Part Design" workbench.

The following syntax is also allowed:

workbench.type.name=

workbench.type.color=

For example:

```
'Part Design'.Pad.Color= 'Sea Green'
```

searches for all objects of type Pad created using the "Part Design" workbench, and of the color Sea Green.

You can also omit certain expressions as follows:

```
'Part Design'.Pad
```

and:

```
.Pad
```

are equivalent. Similarly:

```
'Part Design'.Pad.Color= 'Sea Green'
```

and:

```
type=Pad & Color= 'Sea Green'
```

are also equivalent.

Here are some more examples using other operators described in [Using Operators](#):

```
workbench.type.name= point*
```

```
workbench.type.name!= point*
```

```
workbench.type.name: point*
```

and:

```
workbench.type.color= 'sea green'
```

```
workbench.type.color!= 'sea green'
```

```
workbench.type.color: 'sea green'
```

The following are also allowed:

```
col= 'sea green'
```

```
color='sea green'
```

```
name= * 1
```

```
n= * 1
```

```
type=hole
```

```
t=hole
```

Searching by Color

You assign colors to objects using the Color dialog box Graphic tab, when using the **Edit->Properties** command or the **Properties** contextual command. For a reminder about color names, refer to "[Displaying and Editing Graphic Properties](#)".

To search for an object of a specific color, enter the following command:

```
color: color_name
```

or:

```
col: color_name
```

where "color_name" is the color of the object.

If the name of the color contains a blank (which is the case with most of the colors available), you can type the full name as follows:

```
color: Sea Green
```

You can also surround the blank or the color name with a single quote (by default) like this:

```
color: Light 'Blue
```

or like this:

```
color: 'Light Blue'
```

You can also search for colors using their RGB values. For example:

color: '(200,100,100)'

Searching by Product Properties

You assign properties to products (and parts in products) by right-clicking an element in the specification tree and selecting the **Properties** command from the contextual menu, clicking the Product tab in the Properties dialog box, and setting the properties in the Product frame.

The properties you can search for (the same as those you assigned to the element) are:

- Part Number
- Revision
- Definition
- Nomenclature
- Product Description
- Component Description.

For example, the search queries:

Product Description: *completed*

Product ' Description: *completed*

Product Description ': *completed*

Product Description '= *completed*

search for all elements whose Product Description contains the text "completed".

The property name is not case sensitive. You can also sue the following queries:

product description: *completed*

product ' 'description: *completed*



You can also search by product attributes, for example:

Drafting.Text.'Text String'='my text'

searches for any element containing the text string "my text".

Searching Objects Belonging to a Selection Set

To search for an object belonging to a selection set, enter the following command:

set:selection_set_name

where "selection_set_name" is the name of the selection set.

The strings "name", "type", "color" and "set" are appropriate for the English language only. The message catalog KeyboardInput.CATNls determines the exact syntax for your language.

Searching for Visible or Hidden Elements

You can search for visible elements, or elements hidden in the No Show space using the following syntax:

visibility:visible

vis:visible

and:

visibility:hidden

vis:hidden

Searching for Lines by Thickness or Linetype

You can also search for lines using specific thicknesses or linetypes like this:

weight:

w:

or:

dashed:

d:

When searching for lines with a specific weight, you can specify the weight index like this:

weight: 6,all

to search for all lines which are 2.0000 mm thick.

Searching Using Favorites

You can search for objects using your favorite queries defined via the [Favorites](#) mode. To do so:

favorite= *favorite_query_name* Or favorite:*favorite_query_name*

but you can also enter:

f= *favorite_query_name* Or f:*favorite_query_name*

where "favorite_query_name" is the name of the favorite query. Note that the language is case sensitive.

When you press ENTER, the search results are selected in the specification tree.

You can use any of the special characters such as "=", "!" or "&" and combine them as you wish. For example, let's use the query defined in [Searching Using the Favorites Mode](#). This query searches for any element named "My Sketch". Suppose you want your search to select all elements whose names are other than "My Sketch", you will just have to type:

favorite!=Query_1

or

f!=Query_1

In that case, the search filter is not modified.

You can modify the search filter, just indicate the new filter after the favorite query name in the power input field:

favorite=Query_1, from

will search for any element named "My Sketch" *From Search to bottom* and not *Everywhere* as originally defined in Query_1. The new filter will supercedes the previous filter.

Use of Special Characters in Searches

A name can contain any of the following characters, which also play a special role in the search syntax:

: , & + - () " * . = blank ; ! ' < >

You must use the character ' (apostrophe by default) to surround strings containing special characters and which are to be interpreted exactly.

For example:

name: *!&*!*

searches for all names containing the string &*.

The following line in the KeyboardInput.CATNls resource file :

Quote = ';

specifies the default character for surrounding strings.

However, you may also want to search for names containing the character ' itself. In this case, enter the character twice and make sure you place the two characters outside any string surrounded by the character '. Note also that the character ' can be used when searching for names, types, colors and selection sets.

For example:

```
name: '*'*
```

searches for all names containing a '.

Another example:

```
name=my' favorite' part.'1'
```

is identical to

```
name='my favorite part.1'
```

The above example shows that if a text string does not contain any apostrophe but special characters, it may be surrounded by apostrophes.

Using Operators

The supported operators are: &, +, and - (for AND, OR and EXCEPT respectively) and ().

Blanks are not considered as separators. They may be surrounded by ', but this is not mandatory.

The following examples:

```
name: *wheel'&'door&type:Part
name: *wheel'&'door & type:Part
name: *'wheel&door' & type:Part
name: *'wheel&door'& type:Part
name: *wheel'&'door &type:Part
```

all search for parts whose names end with "wheel&door".

The command:

`name:*wheel'`

searches for names ending with "wheel".

The command:

`name:'wheel*`

searches for names beginning with "wheel".

The command:

`name:wheel1 + name:wheel2`

searches for two objects named "wheel1" and "wheel2".

Upper and Lower Case

When searching for names, if the value is entered in lower case, the case is ignored. If there is at least one upper case character, the case is taken into account.

Priority

There is no priority among operators, but there is a priority in their order of appearance (from left to right).

For example, the query:

`type:Part & name:toto + type:Hole & Color:Black`

is interpreted as:

`type:Part & (name:toto + (type:Hole & Color:Black))`

and thus searches for elements of type "Part" among the objects in the document that are "black holes" or named "toto".

To avoid ambiguity, use parentheses like this:

`type:Part & (name:toto + (type:Hole & Color:Black))`

to obtain the same result.

Using Search Filters

You can use the same search filters (except In List) as with the **Edit->Search...** command, by using the context keywords "all", "in", "from" and "sel":

- **all**: searches the whole specification tree from top to bottom, to find objects created using all workbenches.
- **in**: the search will locate the appropriate elements in the active object and in the workbench you are currently using
- **from**: searches the elements in the active object to the bottom of the tree. For example, in a Part document, both parts and sketches are searched.
- **sel**: if you already selected objects before selecting the **Search...** command, this option searches from the selected objects to the bottom of the tree.

The default is "in". Context keywords must always be placed at the end of the search string, and after the separator ",", like this:

`type:Hole, all`

The separator is the comma (",") for the English, Japanese, Korean and simplified Chinese environments, and the semi-colon (";") for all other supported environments.



Virtual Reality Configurations



About Virtual Reality Support in Version 5
Stereoscopic Viewing
Running a Multipiped Version 5 Session
Working With the Immersive System Assistant
Using Head and Hand Tracking Devices in Version 5
Using the Joystick Control Panel
Navigating in Examine Mode
Navigating in Fly Mode
Installation Requirements

About Virtual Reality Support in Version 5



What is Virtual Reality?

Clicking and dragging your way through a 3D graphics application may be interactive, but it's not virtual reality. It's not Virtual Reality because it's not immersive. "Immersion" (or "presence") is the goal of virtual reality.

Immersion refers to your sense of engagement with the virtual model or environment. When you're immersed, you focus directly on your subject and disregard everything else. In virtual reality, an "immersive" application lets you focus on the task at hand, so you don't deal with a mouse, or the UNIX command line, or a GUI, or the fact that you're actually looking at something made from digital bits instead of physical atoms.

Virtual reality essentially moves the computer out of your way, so you can interact directly with your data or information. Unlike interactive 3D graphics (which can consist of a series of still images coming off disk), virtual reality lets you intuitively manipulate and navigate through a real-time simulation of an object, or a process, or a place.

Where is Virtual Reality technology useful?

Virtual reality adds value to virtually any application where it's vital to experience spatial relationships, and analyze, design, engineer and understand such relationships. Any project in which 3D information must be navigated or closely examined will benefit from virtual reality technology.

If you are working in two dimensions (web design, word processing, ...), you don't need virtual reality.

What are the different types of Virtual Reality configurations?

Virtual reality technology can be recognized by the presence of specific I/O devices which can be organized into five categories, each creating a different impression of immersion. These categories are not mutually exclusive:

- *Simulators*

Simulators use a physical mockup of vehicle with real controls (steering, throttle, pedals, etc.), with which you can navigate through virtual environment, and are ideal for applications such as pilot training, driver training and games. Simulators can also involve multiple participants.

- *Wearable devices*

Wearable devices provide direct, "body contact" input/output to virtual models or environments through such devices as head-mounted displays, boom-mounted displays, data gloves, data suits, haptic feedback systems, motion platforms. These are ideal for digital prototyping, and provide a highly immersive experience. However, they are limited to single users.

- *Desktop devices*

The traditional monitor serves as a window into virtual world, with which you interact using shutter glasses and 3D input. This is a low-cost alternative to wearable devices, and are suitable for scientific data visualization. They are also flexible (easily switch from monoscopic to stereoscopic; easy keyboard access). This range of equipment includes stereoscopic workbenches, tables and desks.

However, desktop devices suffer from the following drawbacks:

- they are based on the single user approach
 - they do not provide a full immersion experience.
- *External devices*

External devices are used to interact with your Version 5 session. Those devices consist of a hand held equipment in which a tracker and a set of buttons have been integrated. Hand tracking allows you to add a virtual hand in the model to perform interactive functions such as selection, command run or navigation.

- *Display systems*

Large virtual models and environments are projected onto flat or curved screens, using such technologies as virtual reality walls (for example, in Silicon Graphics Reality Centers). They can also be projected onto vertical and/or horizontal surfaces in special chambers like "caves" (for example, the Fakespace CAVE) or "walk-in domes".

These devices are the most sophisticated (and most expensive). They are ideal for digital

prototyping, provide high resolution, and allow groups of participants to get involved and collaborate. They also require a lot of physical space.

Which Virtual Reality tools/technologies can be used in Version 5?

In the myriad of tools and configurations available, we can already identify a short-list of hardware useful for navigating through or manipulating CAD data in real time, and capable of providing some level of immersive experience:

- Stereoscopic viewing

Stereoscopic viewing of 3D images is built in Version 5. It can be achieved either in active or passive stereo mode, depending on the display system abilities.

In an active stereo display system, left eye and right eye images are alternatively displayed on the screen at twice the refresh rate. An active pair of glasses with two shutters working in synchronization with the images is needed. Synchronization is often achieved using an infrared emitter: the left eye shutter is closed when the right eye image is displayed, and reciprocally.

In a passive stereo display system, left eye and right eye images are displayed simultaneously on the screen. Image separation is performed by filtering glasses using polarized light, for instance.

Stereoscopic viewing is extremely easy to implement. All hardware manufacturers provide now graphic boards supporting OpenGL quad-buffered stereo on Windows and UNIX based systems. Stereoscopic shutter glasses, such as the CrystalEyes[®] manufactured by StereoGraphics, provide a partially immersive experience of digital mock-up visualization at lowest price investment. Another possibility is to use a StereoGraphics Z-Screen mounted on top of a standard monitor and viewed wearing non-expensive and lightweight passive stereo glasses.

- Head mounted display

Head mounted display (HMD) afford a more immersive experience. Equipped with position-tracking capability, it displays output imagery based upon the position of your head.

To use a head mounted display, you can have either:

- an image generator computing left eye and right eye image separately such as an SGI Onyx workstation. Those are high-end systems providing high performance and high quality 3D viewing

or

- an active to passive stereo converter such as Cyviz XPO2 box. This solution allows you to use an head mounted display on any platform supporting active stereoscopy.

In addition to head mounted display, a standard joystick can be added to the virtual reality configuration, providing a very easy way to navigate in the digital mock-up.

- Projector tables, projection walls, Immersive Rooms

Digital prototyping is served well by using a large projection systems going from the single screen projection table, such as Fakespace Systems ImmersaDesk or Barco Baron, to the multi-projector and multi-sided projection room, such as the FakeSpace Systems CAVE or the Barco I-Space. Version 5 also supports reconfigurable immersive display systems such as the Fakespace RAVE or the Barco MoVE. Those systems are ideal for scale one digital mock-up review or designing large assemblies and facilities.

Displaying the Version 5 on a single screen/single projector display system requires no specific Version 5 service. You just need to plug your machine graphic board video output to the display video input.

When you have a multi-projector display system, for instance as in a TAN Holobench or in a SGI Reality Center, you need a specific hardware and specific Version 5 functionalities. Using a multipipe SGI Onyx workstation, Version 5 can support any kind of immersive environment, that is to say any number of screens and projectors, in passive or active stereo vision mode.

In such environments a better immersion is achieved using head tracking, providing a 3D image depending on the exact user point of view, as well as hand tracking for immersive interaction. To do so, Version 5 supports:

- Trackers such as Polhemus Fastrak[®], Intersense IS-900 and any devices compliant with the Fastrak Protocol
- Hand held devices with integrated tracker and buttons such as Virtual Presence SpaceStick, Fakespace Neowand or Intersense Wand
- 3D Connexion Spaceball and Space Mouse which could dramatically increase your work productivity by helping you manipulating the model with your second hand. Please note that those devices are not specific to virtual reality and can be used in any standard desktop configuration.

What does Version 5 support natively?

The Version 5 infrastructure provides support for:

- stereoscopic viewing: refer to [Stereoscopic Viewing](#) for more information
- multipipe and multithread rendering: refer to [Running a Multipiped, Multithreaded Version 5 Session](#) for more information
- head and hand tracking: refer to [Using Head and Hand Tracking Devices in Version 5](#) for more information.



Stereoscopic Viewing

P2

P3



This section provides background information about stereoscopic viewing.

Computer technology uses stereoscopic viewing to recreate the way we naturally see depth - stereoscopically. Stereoscopic viewing describes how we use both eyes, each with a slightly different perspective, to perceive depth in a physical environment. It delivers the most realistic visual representation possible of complex digital models, giving engineers, architects and scientists the best possible understanding of three-dimensional information, and yielding levels of technical proficiency not available using a typical 3D view.

These images can be perceived by a user wearing special glasses which continuously transmit separate images to the left and right eyes, creating a view of computer or video-based objects that have depth, perspective and presence in three-dimensional space.



We do not intend to describe here all the possible hardware configurations which support stereoscopic viewing for Version 5. Consider the examples of hardware configurations mentioned in this section as no more than that: just examples.

What Do You Need?

Stereoscopic viewing is possible on both entry-range and high-end configurations: you do not automatically need expensive equipment to enter the realm of stereoscopic viewing.

Entry-Range Configurations

On entry-range configurations, you need at least a graphics board supporting stereoscopic viewing on your platform, and a set of special glasses.

The CrystalEyes[®] range of glasses (designed by StereoGraphics, Inc.) is an example of the type of special glasses supported allowing you to benefit from stereoscopic viewing capabilities.


Many graphics boards are supported. For detailed information about supported hardware configurations if you are running Windows, and general information about StereoGraphics, Inc. products, browse the following Internet site:

<http://www.stereographics.com/boards/brd-chrt.htm>

In case you want to use a standard machine with only one graphic board, two video inputs (one per eye) are required to manage a stereoscopic display system. If you have a machine with only one graphic board providing a single video output, the Cyviz XPO2 is a solution.

The Cyviz XPO2 is a small electronic box converting one active stereo video input into the two corresponding plain video outputs (one per eye). A typical application would be a projection wall with passive stereo (two projectors) or a head mounted display.



You can click the thumbnail  if you want to display a full-size picture of the XPO2.

For detailed information about XPO2, please browse the following internet site :

<http://www.cyviz.com>

High-End Configurations

High-end configurations typically involve not only specific graphics boards and special glasses, but also a whole range of high-quality, immersive, stereoscopic display platforms from vendors such as FakeSpace Systems, which allow you to manipulate, assemble, and disassemble virtual mechanical objects while navigating through the entire digital mock-up.

For detailed information about supported hardware configurations, and general information about FakeSpace Systems products, browse the following Internet site:

<http://www.fakespacesystems.com/>

http://www.barco.com/projection_systems/

How to setup stereoscopic viewing in Version 5?

This section is dedicated to stereoscopic viewing on systems that use one graphic board supporting OpenGL quad buffered stereo.



- 1.** Set up the appropriate hardware configuration.

- 2.** Set up the graphic adaptor configuration for stereoscopic display.

This step depends on hardware and operating system configuration. Some systems require administrator privileges to change the graphic adaptor display mode.

You are required to determine the height, width and frequency characteristics of the graphic interface.

On IRIX:

With root privilege, from the command line, you can use the setmon command, for instance:

```
$ ls /usr/gfx/setmon -x 1024x768_96s
```

For this to be taken into account, restart the graphic subsystem:

```
$ (/usr/gfx/stopgfx;/usr/gfx/startgfx)&
```

On SGI Onyx2 or SGI Onyx 3000 system, you should rather use the **ircombine** command. For detailed information, please refer to the SGI documentation.

On Solaris:

On Solaris 8 use the fbconfig command, first to know the available configurations:

```
$ /usr/sbin/fbconfig -res ?
```

Choose a configuration, for instance 1280x1024x114s, then use the fbconfig command once again:

```
$ /usr/sbin/fbconfig -res 1280x1024x114s
```

Logout to restart the X server.

On previous Solaris versions, the configuration command may depend on the kind of graphic adaptor you have: ffbconfig for Creator 3D, afbconfig for Elite and SUNWifb_config for Expert 3D.

On AIX:

On AIX systems, you can use the smit utility:

- log as root
- run smit
- choose "Devices"
- choose "Graphic Displays"
- choose "Select the Display and Resolution refresh Rates", then the correct graphic board if more than one are on the system
- choose a stereo visual. The "list" button provides a list of all available graphic modes. Stereo modes are at 120Hz, for instance 1024x768@120Hz.

On HP-UX:

With root privilege, use the setmon command to know the available configurations:

```
$ /opt/graphics/common/bin/setmon -r
```

This gives a table on which each line corresponds to an available graphic mode. To choose a graphic mode, type:

```
$ /opt/graphics/common/bin/setmon -s n
```

where "n" is the entry number of the chosen graphic mode.

On Windows based platforms

Stereo activation can be found in "Display properties" (select the "properties" contextual command in your desktop background). Depending on the graphic adaptor, you may need to have administrator privilege to access the stereo setting.

The location of this setting varies depending on the graphic driver vendor. If it is not directly available in the vendor. If it is not directly available in the vendor tab page, look for "advanced configuration" in the vendor tab page or in the "settings" tab page. In some cases, you may need to reboot the machine to activate the new mode.

3. Start a session.
4. Select the **Tools->Options->General->Display->Visualization** command.
5. Click the Devices tab, and set the Stereo option to ON to enable stereoscopic viewing.

Note the following platform-specific restrictions when enabling stereoscopic viewing:

- on AIX: you lose colors by passing from 24-bit to 12-bit colors using the GXT800 graphics board
- on SGI: performance decreases by approximately 10%.

6. Exit your session to save your settings, then restart.

7. Display the Stereoscopic viewing dialog box as follows:

- enter the following command in the power input field:

c:Stereoscopic

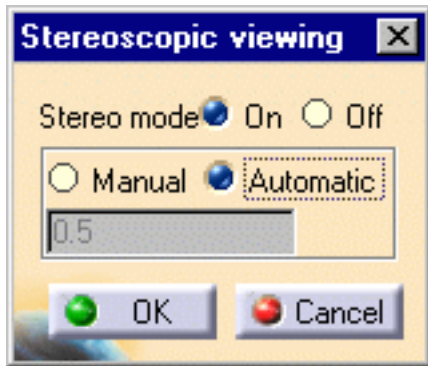
- or, select the **Tools->Customize...** command, select the Commands tab, select the "All commands" item from the "Categories" list, then select the "Stereoscopic..." command. You can then add the command to a toolbar for easy access as explained in "[Customizing a Toolbar by Dragging and Dropping](#)". Select the command once it has been moved to the toolbar
- or select the **View->Commands List...** command then choose the "Stereoscopic..." command from the list.

The Stereoscopic viewing dialog box looks like this:



By default, stereoscopic viewing is disabled (Off).

8. Set the Stereo option to On to enable stereoscopic viewing.



Note that visualization performance will be impaired if you set the Stereo option to ON.

9. Set either "Manual" or "Automatic" mode.

This mode sets the distance between your eyes when using stereoscopic viewing. The default mode is "Automatic".

Automatic mode

When working in Examine mode, you should set the eye gap to "automatic". Use this setting if you are viewing an object that in real life you could reasonably manipulate using your hands. This setting is suitable for working in confined spaces requiring an accurate perception of depth, in which you focus on the rotation point.

When automatic mode is active, the eye-distance is adjusted automatically depending on the zoom factor.

In automatic mode, the line of sight converges on the focal point. If the focal point is far away, and objects are located before the focal point, these objects will be viewed in a fashion more precise than in real life, so viewing results are unrealistic in this case.

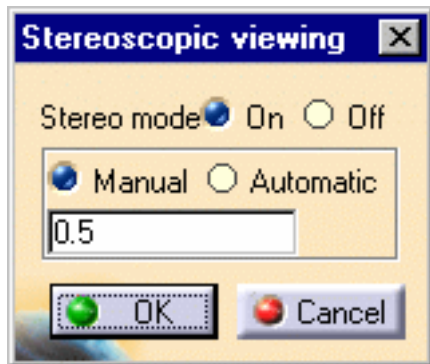
Manual mode

Manual mode provides you with precise control over the distance between your eyes, so that you can adapt your field of vision to the working context.

The value you enter will be in millimeters.

This mode is particularly suitable for working in Walk or Fly mode, when working on large-scale assemblies or industrial plants requiring a wider field of vision. If you need to perceive a high degree of depth, you should set the distance between your eyes accordingly.

In this mode, the line of sight is parallel.



Running a Multipiped Version 5 Session



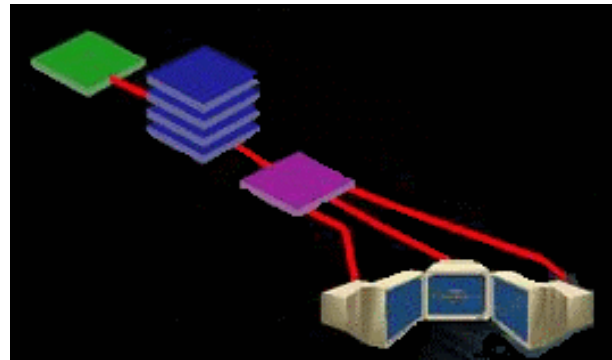
In [About Virtual Reality Support in Version 5](#), we discussed the different types of virtual reality configurations available. However, intensive use of virtual reality technology leads to loss of performance if you are using only one graphics pipe.

What is a "pipe" and what is the difference between a single and multipipe system?

A "pipe" in this context refers to the graphics board associated with each single window into an application. It also refers to the data that is placed by the application which owns the pipe.

You can also have a single pipe feeding three display channels like this:

Nothing prevents you from using virtual reality configurations with just one pipe, the problem is that the overhead of running multiple applications in a single piped system can result in very poor performance.



In a "multipipe" system, you can have several graphics boards, and each graphics board feeds a different display channel. Each application in a "multipipe" system has its own distinct pipe to the graphics hardware. There is a price to pay in memory usage and a very slight price to pay in computer overhead to manage the multiple pipes, but the overall positive effect on system performance is dramatic.

Performance can be enhanced even further if you are running workstations using multiple processors, which allow "multithreading" of applications.

Note that:

- the Version 5 infrastructure for UNIX platforms provides support for 16 windows maximum
- one window/pipe can be assigned to two threads (2 CPUs).



A typical configuration is an SGI Onyx2 or SGI Onyx 3000 workstation using four graphics pipes. You run a main Version 5 session using one pipe from a workstation, and create three secondary windows (each one using its own graphics pipe) for projection onto a Reality Center-type projection screen, or inside a virtual reality CAVE.



1. Log onto the Onyx workstation.

2. Run the following command to display graphics subsystem information about your configuration:

```
/usr/gfx/gfxinfo
```

This command displays information (display name, resolution, etc.) about each graphics pipe like this:

```
Graphics board 0 is "KONAL" graphics.  
Managed (":0.0") 1280x1024  
...  
Graphics board 1 is "KONAL" graphics.  
Managed (":0.1") 1280x1024  
...  
Graphics board 0 is "KONAL" graphics.  
Managed (":0.2") 1280x1024  
...  
Graphics board 1 is "KONAL" graphics.  
Managed (":0.3") 1280x1024  
...
```

3. Synchronize the graphics pipes with the video channels in your configuration by running the program:

```
/usr/gfx/ircombine
```

Refer to your IRIX documentation for more information about the `ircombine` program.

4. Export one of the following environment variables:

```
export CATMPConfig=path/myconfigfile
```

or

```
export CATMPKConfig=path/myconfigfile
```

where "*path*" is the path of a directory and "*myconfigfile*" is the name of a configuration file which you must create and edit to set up the secondary windows and assign them to different graphics pipes and displays.

Version 5 supports OpenGL Multipipe SDK technology (MPK) as well as an older set of SGI multipipe services. The MPK library is developed by SGI to manage multi-channel display configurations. Version 5 `mpconfig` command switches on and off the Version 5 multichannel support. The prerequisite is that one of the environment variables, i.e. `CATMPKConfig` or `CATMPConfig`, is valued to a configuration file path before running the application:

- if `CATMPConfig` is valued, the command activated the old set of multipipe services
- if `CATMPKConfig` is valued, the command activates the MPK support.

Using MPK Version 5 provides powerful support for many virtual reality configurations:

- active or passive stereo
- decompositions, such as Stereo (Eye), DPLEX, 2D
- all multi-screen/multi-projector configurations such as TAN Holobench, Fakespace CAVE, etc.
- viewpoint tracking on all screens.

Note: in order to use MPK, you need to download the corresponding library. For detailed information, browse the following internet site:

<http://www.sgi.com/software/multipipe/sdk/>

When turned on using the **mpconfig** command, the multipipe support displays an information window (named "Multi Pipe started") in which you just have to click OK to proceed. Some additional windows will then appear on the managed piped and the Version 5 model will be displayed across all windows.



These two variables are not compatible and if you export both of them, the CATMPConfig variable supersedes the CATMPKConfig variable.

5. Edit the configuration file.

If the display **":0.0"** is used for the main Version 5 session, the syntax of the following file shows how to project a session using three separate pipes and the CATMPConfig environment variable :

```

pipe                // First pipe declaration
{
name ":0.1"         // Display name
window             // First window declaration
{
x 0                //Horizontal position (in pixels)
y 0                //Vertical position (in pixels)
width 1280         //Window width (in pixels)
height 1024        //Window (in pixels)
reference          //Use width and height of this window to compute all
                    size parameters
wall_position [0,0] //Define the wall position (in pixels); the center of the main window is
0,0
}
}

pipe                // Second pipe declaration
{
name ":0.2"         // Display name
window             // Second window declaration
{

```

```
x 0 //Horizontal position (in pixels)
y 0 //Vertical position (in pixels)
width 1280 //Window width (in pixels)
height 1024 //Window height (in pixels)
```

```
wall_position [1280, 0]
}
}
```

```
pipe // Third pipe declaration
{
name ":0.3" // Display name
window // Third window declaration
{
x 0 //Horizontal position (in pixels)
y 0 //Vertical position (in pixels)
width 1280 //Window width (in pixels)
height 1024 //Window height (in pixels)
```

```
wall_position [-1280, 0]
}
}
```

6. Save your changes to the configuration file, then start your main Version 5 session.

7. In your main session, run the command:

c:MPConfig

to start multipiping. This creates the three empty secondary windows whose display name, size and position you defined in the configuration file.

If you are using a Reality Center-type projection screen, for example, the three windows will be displayed side-by-side.

8. In your main session, run the command, select the command **View->Full Screen**.

The video output displayed inside your main session is now projected coherently across the three windows.

9. In your main session, run the command:

c:MPConfig

again to stop multipiping.



Note that you can use stereoscopic viewing from within a multipiped session.

Note also that in CAVE configuration, you can increase the specular lighting accuracy by checking the "Enable OpenGL local viewer lighting" option in **Tools->Options->General->Display->Performances**. Be aware that this may lead to a loss of performance.

Calibrating the multi-channel support

First of all, let's remind us of the SGI pipe and channel architecture.

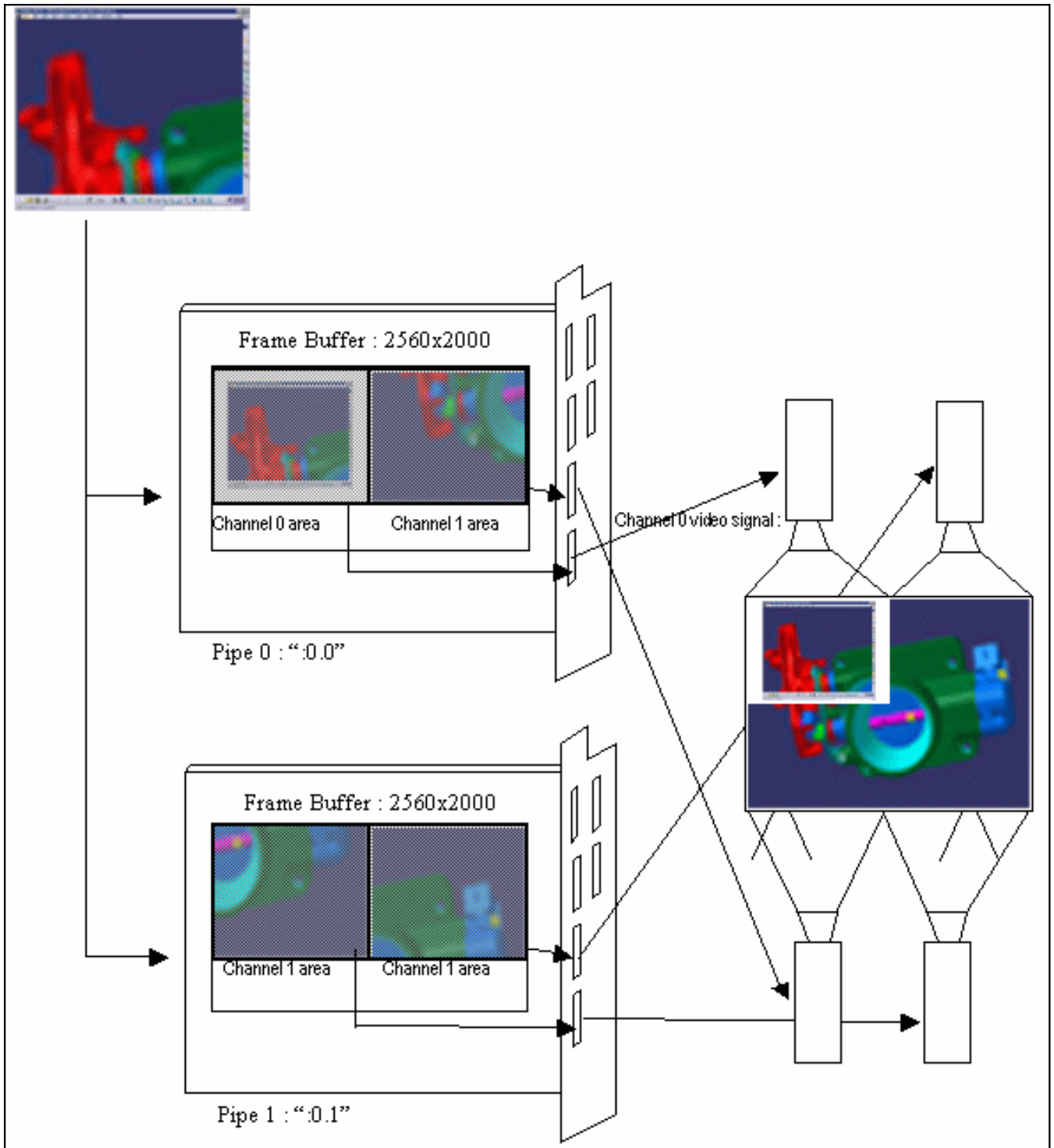
An SGI machine as an Octane or an Onyx may host several graphic boards, named pipes. A same application can render images on all of these pipes. A pipe has a frame buffer in which the application write the images they compute. More precisely, each application writes images in windows that are sub-parts of the frame buffer.

On the video output side, the system reads the pipes frame buffer and generate video signals. There is one video signal per active channel. Each pipe may host several channels. Each channel is positioned on a part of a pipe frame buffer, and a video signal generated display this particular part.

The important characteristic of the system is the number of pipes, their frame buffer size (called "managed area"), the number of channel per pipe, their corresponding frame buffer sub area and the video signal format they output.

You get the system information by typing `/usr/gfx/gfxinfo` in a shell. You can modify your system configuration using the "setmon" or "ircombine" commands. They enable you to change the pipe-managed area size, the activated channels on each pipe, their frame buffer part as well as their video format. Report to the SGI documentation to use these commands.

The pipes are usually named ":0.0" , ":0.1" , ":0.2" ..., unless you have several X Servers, in which case they are named ":0.0", ":0.1" ... on X Server 0, ":1.0", ":1.1",..., on X server 1



On the scheme above, the system has two pipes and six channels per pipe. Only two are activated on each pipe. The pipe-managed areas are 2560x2000.

The Pipe 0, channel 0 is associated to an area of size 1280x1024, and located at the coordinate (0,0).
The Pipe 0, channel 1 is associated to an area of size 1280x1024, and located at the coordinate (0.5,0).
The Pipe 1, channel 0 is associated to an area of size 1280x1024, and located at the coordinate (0,0).
The Pipe 1, channel 1 is associated to an area of size 1280x1024, and located at the coordinate (0.5,0).

The channel video format are 1280x1024 at 96Hz stereo for instance.

Version 5 renders 3D images in four windows. Two of them are on pipe 0, and the other two on pipe 1. Among these windows, one is particular: this is the main document window which appears within the application frame. The other windows are called "auxiliary windows". If the user goes to the "FullScreen" mode, a complete image of the model will appear on the screen.

All the information Version 5 needs for such a configuration are in a configuration file (refer to [SGI MPK support](#)). However, the system should also be correctly set up using the "**ircombine**" or "**setmon**" commands.

Also, since this system outputs active stereo, the pipes need to be "genlocked". This means that their video refresh are synchronized. This is necessary for the stereo glasses to work correctly. You should refer to the SGI support for genlocking your system.

For instance, to set up the system, you can run the following script in a shell. You need the administrator privileges. This script will log you out and may badly affect the system if there is an error in it.

```
echo "Pipe 0 INTERNAL (master). Setting ..."  
/usr/gfx/ircombine \  
-target :0.0 \  
-destination eeprom \  
-global syncsource=INTERNAL,size=2560x2000 \  
-channel 0 format=1280x1024_96s,sync=RGB,sourceloc=0+0 \  
-channel 1 format=1280x1024_96s,sync=RGB,sourceloc=1280+0  
echo "done."  
  
echo "Pipe 1 EXTERNAL (slave). Setting ..."  
/usr/gfx/ircombine \  
-target :0.1 \  
-destination eeprom \  
-global syncsource=EXTERNAL,syncformat=1280x1024_96s,size=2560x2000 \  
-channel 0 format=1280x1024_96s,sync=RGB,sourceloc=0+0 \  
-channel 1 format=1280x1024_96s,sync=RGB,sourceloc=1280+0  
echo "done."  
echo  
  
(/usr/gfx/stopgfx;/usr/gfx/startgfx)&
```

There are two kind of configuration file you can use to deal with multi-channels configurations. We only speak here about the MPK config file. To run V5 with such a configuration file, type in a shell:

```
export CATMPKConfig={ complete path of the config file}.
```

then, in this same shell, run CATIA :

```
/usr/DassaultSystemes/Bn/irix_a/command/catstart -env {EnvName} -direnv {environment path name}
```

where "n" is the release number.

The "**Mpconfig**" command will use the configuration file pointed by the CATMPKConfig environment variable. To have a full understanding of the MPK config file format, please refer to the MPK documentation. Only a standard display configuration will be taken as an example and detailed here.

Let's assume we have a cubic immersive space such as a Fakespace CAVE, or a Barco Cubic Immersive Space. This system has four walls. All of them are square. The length of each square side is 3 meters long. There are four projectors displaying active stereo (one projector per wall).

The machine has two pipes and six channels per pipe. It is set (ircombine) so that:

- The pipe-managed areas are 2560x1024
- Pipe 0 channel 0 goes to the front wall. Its size is 1280x1024. Its location is (0,0)
- Pipe 0 channel 1 goes to the right wall. Its size is 1280x1024. Its location is (1280,0)
- Pipe 1 channel 0 goes to the left wall. Its size is 1280x1024. Its location is (0,0)
- Pipe 1 channel 1 goes to the floor. Its size is 1280x1024. Its location is (1280,0)

The mpk file writes as follows:

```
global
{
  MPK_WATTR_PLANES_ALPHA 1
  MPK_DEFAULT_EYE_OFFSET 57
  MPK_WATTR_HINTS_STEREO 1
  MPK_DATTR_QUADSTEREO_HEIGHT 768
  MPK_DATTR_QUADSTEREO_WIDTH 1024
}
config
{
  name "Config"
  mode stereo
  mono [ "", none ]
```

This is the global section where you set some global options.

Always specify the MPK_WATTR_HINTS_STEREO option when using stereo. Set thMPK_DATTR_QUADSTEREO_HEIGHT and MPK_DATTR_QUADSTEREO_WIDTH to your manage area size when using stereo.

Set the "mode" to "stereo" or "mono" to activate or not the stereoscopic rendering.

Declare the pipes you are using and specify their name.

In each pipe section, declare the windows Version 5

```

stereo [ "", quad ]

pipe
{
  display      ":0.0"
  window
  {
    name       "left window"
    viewport   [0, 0, 1, 1 ]
    channel
    {
      name     "left view"
      viewport [ 0., 0., 1., 1. ]
      wall
      {
        bottom_left [-1828.8, -1524, 1828.8 ]
        bottom_right [-1828.8, -1524, -1828.8 ]
        top_left     [-1828.8, 1524, 1828.8 ]
      }
    }
  }
}

window
{
  name       "front window"
  viewport   [0, 0, 1, 1 ]
  channel
  {
    name     "front view"
    viewport [ 0., 0., 1., 1. ]
    wall
    {
      bottom_left [-1828.8, -1524, -1828.8 ]
      bottom_right [1828.8, -1524, -1828.8 ]
      top_left     [-1828.8, 1524, -1828.8 ]
    }
  }
}
}

```

will manage on these pipe (including the main document window). The main document window should always appear first in its pipe. If your run Version 5 in pipe X, the main document window will use the paramaters of the window declared first in the pipe X section of the configuration file.

The window viewport specifies the window size and position within the pipe frame buffer. The figures correspond to a ratio of the managed area size : [x,y,width.height].

You should always declare one channel per window, and only one (Version 5 only support on channel per window). The term channel may be confusing since it does not have the same meaning as the usual pipe channels. Just always declare the channel viewport as [0,0,1,1]. Then, specify the coordinates of the wall corners. This window image will then be supposed to be projected onto the corresponding wall, fitting exactly the rectangle described by the coordinates.

The coordinates should be expressed in the MPK reference frame. Use the same unit as the unit.

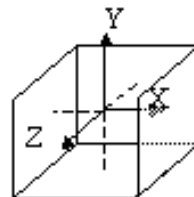
The default eye position is defined by this reference frame. This means that if you use the "MPConfig" command without head tracking, the point of view will be computed for a user standing with its head at this reference frame origin, looking along its -Z axis, and with its right on the positive X axis side.

Since it is usually better to define the system front wall as the one naturally seen by the default head position, the MPK reference frame is chosen so that this front wall is on its -Z axis side.

Bear also in mind that the interaction with the model is only possible within the main window. In particular, this is the case of the viewpoint manipulation with the mouse.

Thus, the front wall is usually set as the main document window.

Here, we get :



```
}
pipe
{
  display      ":0.1"
  window
  {
    name        "right window"
    viewport    [0, 0, 1, 1 ]
    channel
    {
      name      "right view"
      viewport  [ 0., 0., 1., 1. ]
      wall
      {
        bottom_left [1828.8, -1524, -1828.8 ]
        bottom_right [1828.8, -1524, 1828.8 ]
        top_left     [1828.8, 1524, -1828.8 ]
      }
    }
  }
}
window
{
  name        "floor window"
  viewport    [0, 0, 1, 1 ]
  channel
  {
    name      "floor view"
    viewport  [ 0., 0., 1., 1. ]
    wall
    {
      bottom_left [-1828.8, -1524, 1828.8 ]
      bottom_right [1828.8, -1524, 1828.8 ]
      top_left     [-1828.8, -1524, -1828.8 ]
    }
  }
}
}
```



Working With the Immersive System Assistant



In this task, you will learn how to access and use the Immersive System Assistant workbench in order to configure your Virtual Reality devices and display system.

Within this workbench, the various elements (screens, trackers, etc.) any reality center hardware installation is made of are graphically represented and this visual feedback enables you to create, edit and delete very easily the necessary data.

These data will then be used to generate automatically an XML configuration file via a command provided by the Immersive System Assistant workbench, thus avoiding a "painful" manual edition.

From V5R13 onwards, all drivers and brokers supported by Version 5 are threaded which means that there is no need to run external processes anymore.

In addition to that, all the necessary MPK display configuration data (including those related to the threaded brokers and drivers) are now grouped together into a [single configuration file](#) saved in XML format. These data will then be used when starting up your Version 5 session to run the broker and the driver automatically with the appropriate configuration.

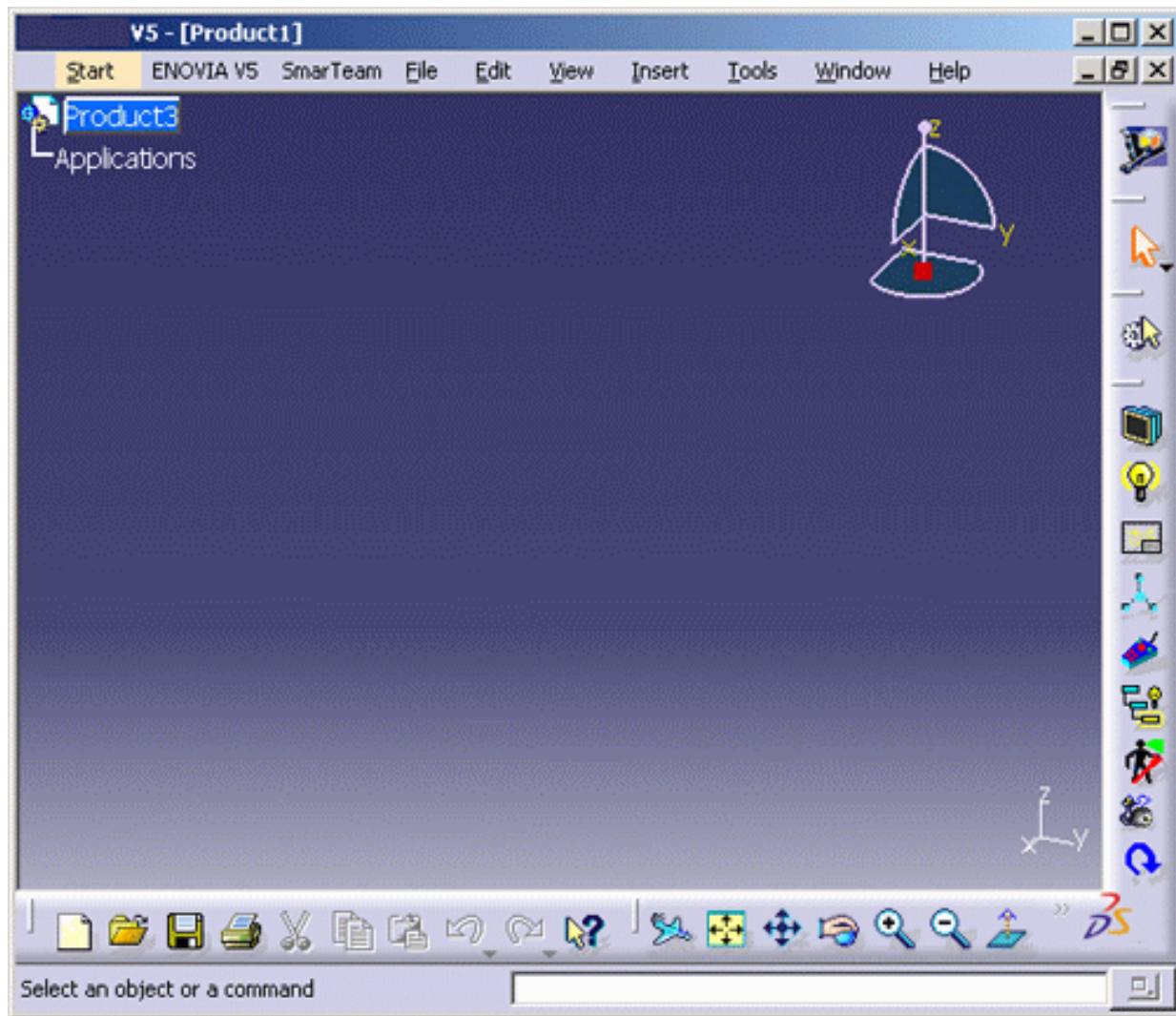
However, note that you can still run drivers and brokers manually as before if needed, for instance when using external drivers or when running a driver on a remote machine.

Real Time Rendering 2 (RTR) users can access and use the Immersive System Assistant workbench. However, a P3 licence is required to be able to use tracking stations and trackers.



1. Access the Immersive System Assistant workbench by selecting the **Start** menu then **Infrastructure-**


> **Immersive System Assistant** . The Immersive System Assistant workbench opens:

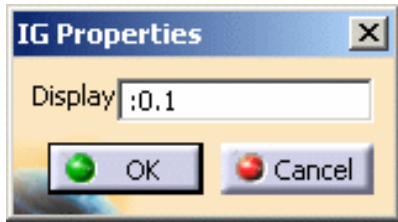



This workbench lets you create and manipulate several types of objects:

- graphic board (or IG for "image generator")
- window
- view (either screen or camera)
- head
- hand picking device
- tracking station
- tracker.

Note that the head, the hand picking device and the trackers are automatically managed and thus, cannot be deleted.

2. Click the **New IG**  icon to create a graphic board (you can modify the default name by entering a new name in the Display field):




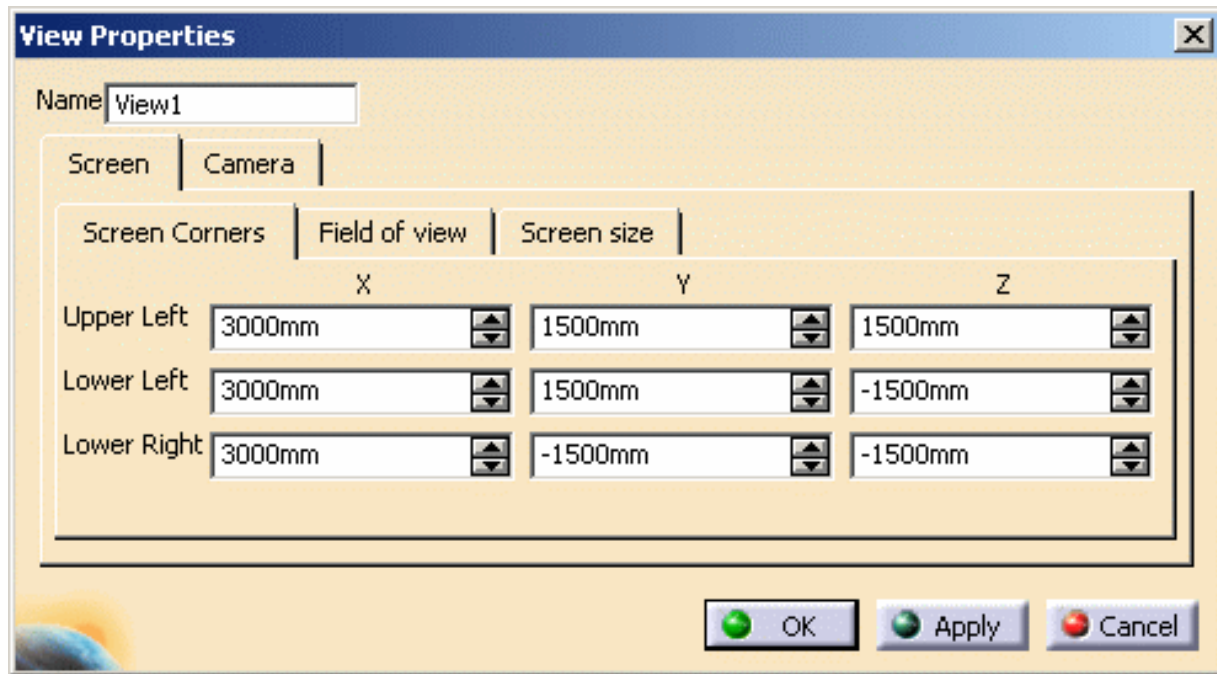
3. Click the **Toggle the plan view on**  icon to display the graphic board symbol in the current window.

You can then hide the graphic board by clicking again this icon.

4. Repeat steps 2 and 3 to create another graphic board and display it in the current window. The geometry area now looks like this:



5. You will now create a rendering specification by clicking the **New View**  icon. This opens the View Properties dialog box which lets you define the type of view you wish to use as well as its properties:



You can choose between a screen view or a camera view:

- screen: use it when the viewpoint to render is defined by the user moving in front of a fixed screen
- camera: use it when the viewpoint to render is attached to the head of the user, like a camera (such as a head-mounted display).

Screen View

1. Click the tab you will use to set the screen properties, either:

- **Screen Corners**
Enter the coordinates of three screen corners (*upper left* corner, *lower left* corner and *lower right* corner)
- **Field of view**

Screen Corners	Field of view	Screen size	
H fov	53.13deg	V fov	53.13deg
H shift	0deg	V shift	0deg
Head	0deg	Pitch	0deg
Distance	3000mm		

The *H fov* and *V fov* fields represent the horizontal and vertical field of view (in degrees), respectively.

The *H shift* and *V shift* fields lets you translate the screen horizontally and vertically.

The *Head* field defines the horizontal rotation of the screen around the head.

The *Pitch* field defines the vertical rotation of the screen around the head.

The *Distance* field defines the distance (in millimeters) to the head.

- **Screen size**

Screen Corners	Field of view	Screen size	
Width	3000mm	Height	3000mm
Head	0deg	Pitch	0deg
X		Y	Z
Center	3000mm	0mm	0mm

The *Width* and *Height* fields correspond to the screen width and height, respectively.

The *Head* field defines the horizontal rotation of the screen around the head.

The *Pitch* field defines the vertical rotation of the screen around the head.

The *Center* area lets you set the coordinates of the screen center.



Once you entered values in a tab, the values of the two other tabs are automatically updated accordingly. You can then pass from one tab to another very easily.

Camera View

1. Click the **Frustum** tab:

"Frustum" is a truncated pyramid representing the angular definition of the field of view for a distance equal to 1. Enter the desired values in millimeters for Left, Right, Top and Bottom.

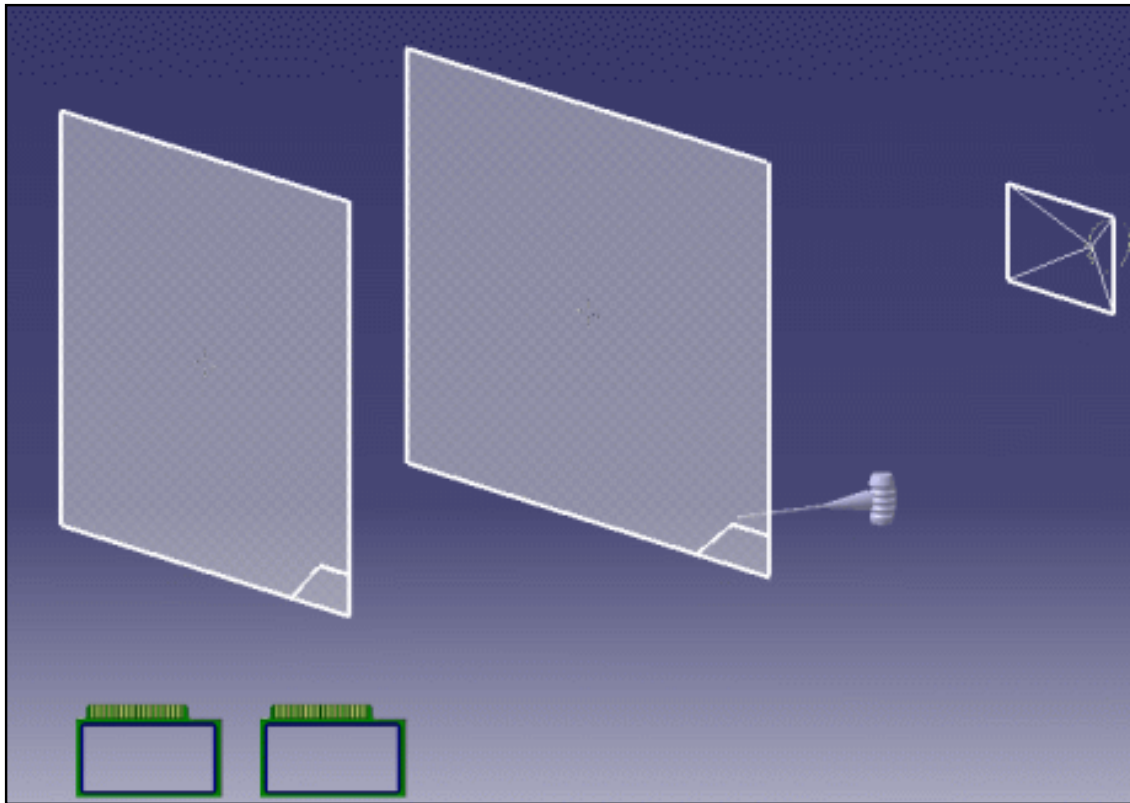
2. Click the **Field of view** tab:


The *H fov* and *V fov* fields represent the horizontal and vertical field of view (in degrees), respectively.
 The *H shift* and *V Shift* fields lets you translate the screen horizontally and vertically.

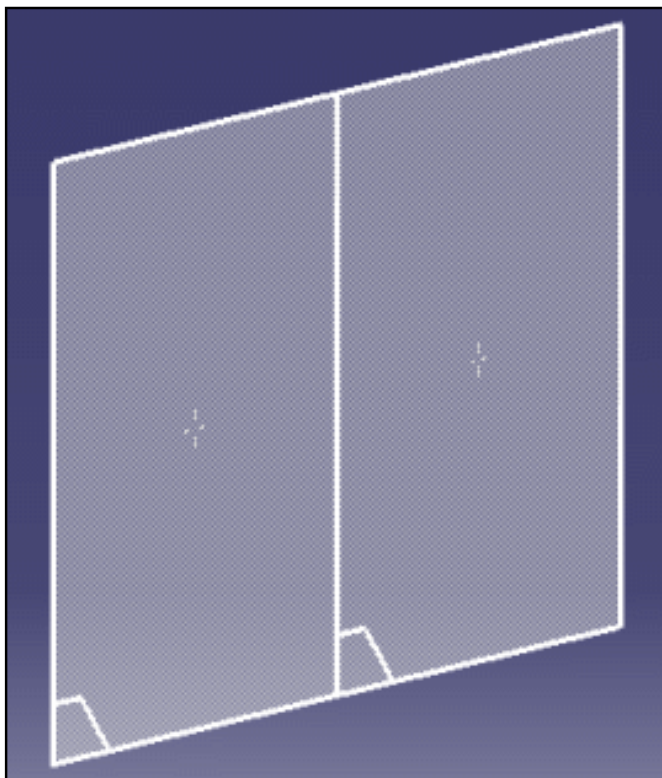
6. Click **OK** when satisfied with your screen definition.
7. Repeat steps 5 and 6 to create another screen view.

Note: you can modify the screen properties at anytime by double-clicking the screen in the geometry area or in the specification tree.

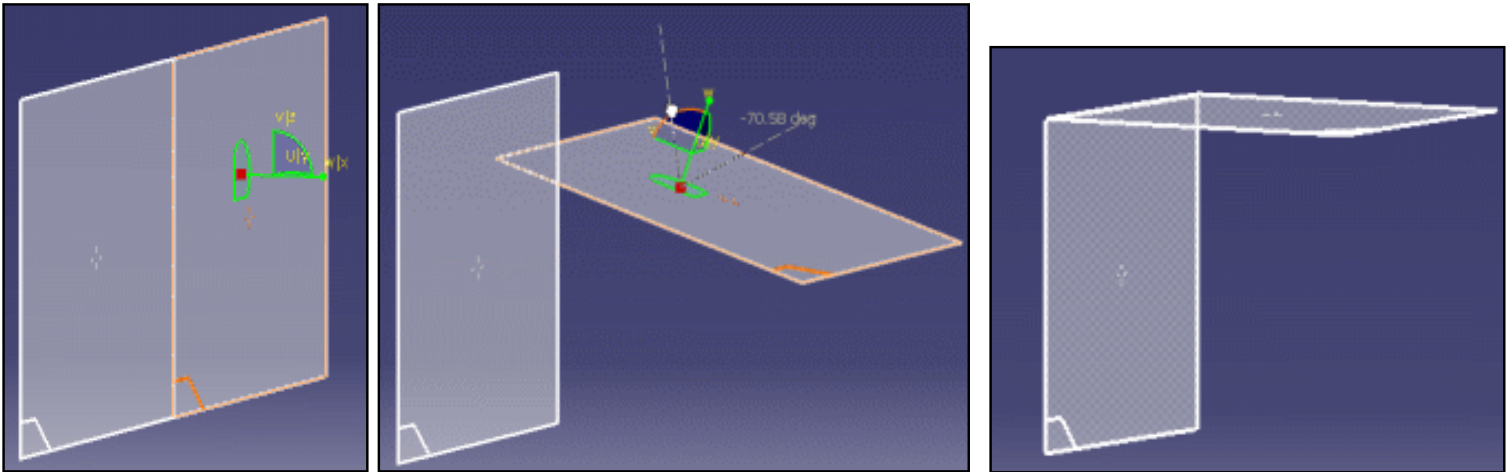
The result looks something like this (depending on the values you enter):



8. Click the **Snap**  icon to snap one of the screens to the other. To do so, select an edge of the screen to be snapped then an edge of the other screen:




You can also snap the 3D compass to a screen view to manipulate it very easily as shown below:

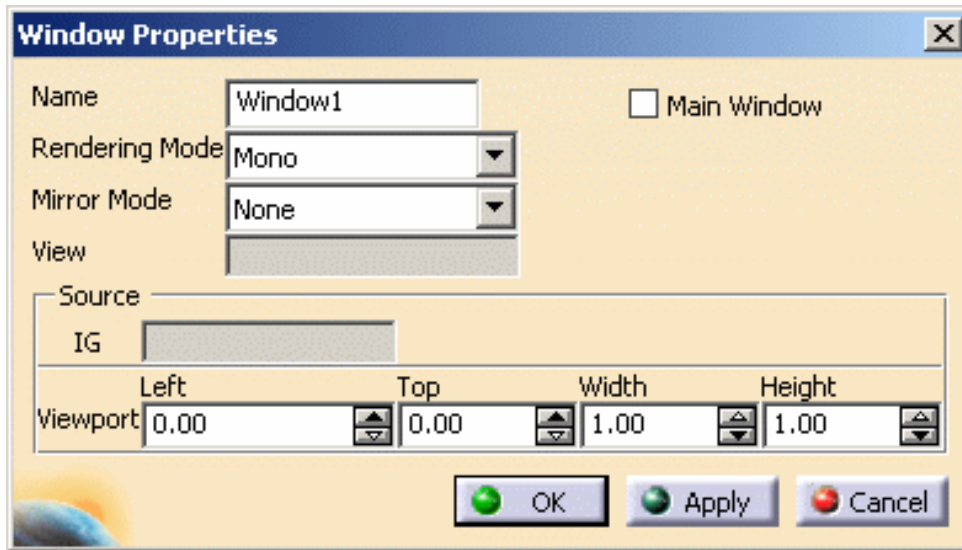


Note that the compass cannot be snapped to camera views.

The next step is to render the specifications in a window.

A window can be defined as an OpenGL rendering zone for a defined viewpoint and is located in the graphic board frame buffer. The specification of the viewpoint to render in a window is called a "view". As a consequence, each window points to a view, either a [screen view](#) or a [camera view](#) depending on the viewpoint to render.

9. Click the **New Window**  icon to open the Window Properties dialog box:



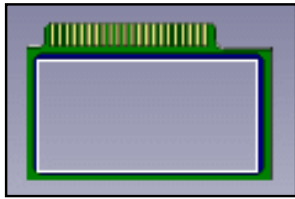
10. Give a name to the new window (or leave the default name) and set it as your referenced window by clicking the *Main Window* option.

The main window is the window which is always visible in Version 5 (the other windows being created "on the fly" when you run the **mpconfig** command).

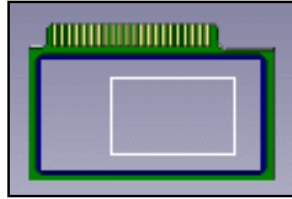
11. Select the associated graphic board in the geometry area or in the specification tree. The *IG* field will be automatically updated accordingly.
12. Select the view the window will point to. The *View* field will be automatically updated accordingly.
13. In the Viewport area, indicate the portion of the graphic board you are going to use. To do so, enter the desired coordinates in the *Left*, *Top*, *Width* and *Height* fields.

0.00 means the view will be located in top left corner

1.00 means the view will be located in bottom right corner.

**Default window**

Left=0.00
Top=0.00
Width=1.00
Height=1.00

**User-defined window**

Left=0.33
Top=0.15
Width=0.57
Height=0.68

14. Specify the *Rendering Mode*:

- Mono: only the cyclop viewpoint is rendered
- Left and Right eye: these modes are used for passive stereo display systems in which left eye and right eye images are displayed simultaneously on the screen. Image separation is performed by filtering glasses using polarized light, for instance
- Active Stereo: left eye and right eye images are alternatively displayed on the screen at twice the refresh rate. An active pair of glasses with two shutters working in synchronization with the images is needed. Synchronization is often achieved using an infrared emitter: the left eye shutter is closed when the right eye image is displayed, and reciprocally.

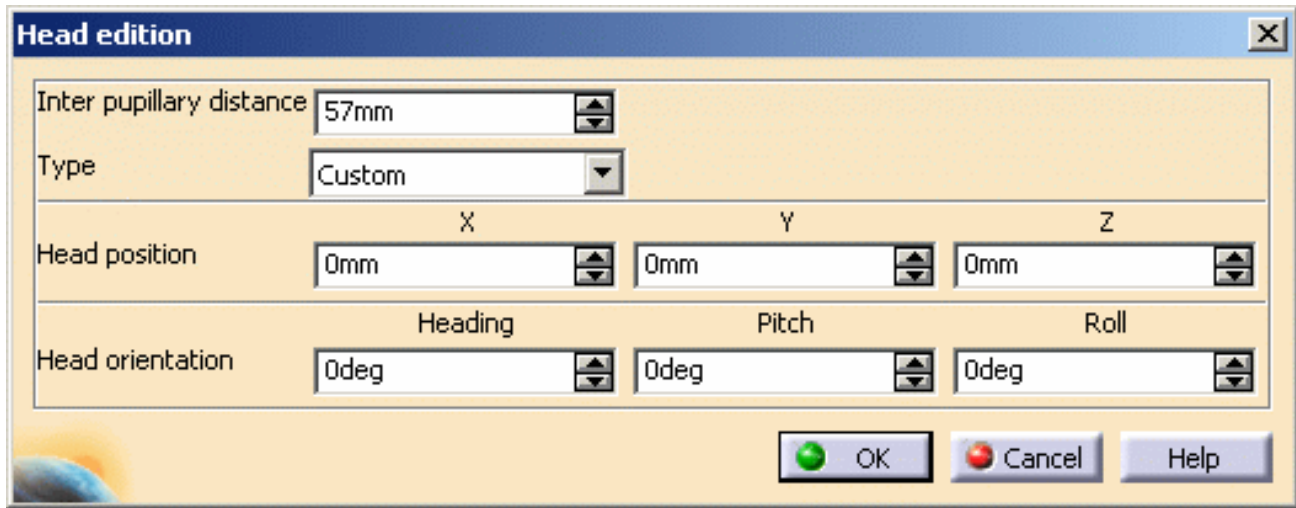
15. Use the *Mirror Mode* field if you want to create a symmetry and define the axis with respect to which the reflected image should be inverted.

Note that you can also create a symmetry using the projector.

16. Click **Apply** then **OK** to validate and close the dialog box.

Now let's go to modifying the head properties!

17. Double-click either the head in the geometry area or the Head item in the specification tree to open the Head edition dialog box:



- indicate the distance (in millimeters) between left eye and right eye in the *Inter pupillary distance* field
- the *Type* field is relevant only when working with devices such as trackers
- define the *Head position* (in millimeters) along X, Y and Z axes
- define the head orientation in the *Heading* field
- set the vertical rotation of the head in the *Pitch* field
- set the horizontal rotation of the head in the *Roll* field.

The head position and orientation you define in this dialog box correspond to the default head position and orientation relative to the screens when the head is not tracked.

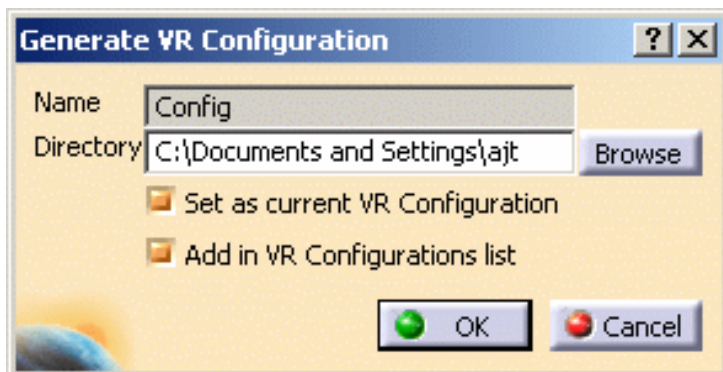
18. Click **OK** to validate.

If you do not use trackers, it is now time to generate your configuration file.

For those who use trackers, a few additional operations need to be carried out before generating the configuration file. Skip the steps below and jump to step [24](#).



19. Click the **Configure VR System**  icon. The following dialog box appears:



The Generate VR Configuration dialog box lets you:

- choose the directory where the configuration file will be saved. You can either enter a path directly in the field or click the **Browse** button then navigate to the desired directory
- set the configuration as the current one by clicking the "Set as current VR Configuration" option: this means that this configuration will be used to run the driver and the broker when restarting your session
- add the configuration to the list of configurations displayed in the **Tools->Options->General->Devices and Virtual Reality->Calibration** tab.

Note that the configuration name cannot be modified in this dialog box. To modify the name, double-click the configuration in the specification tree then, in the VR Configuration Properties dialog box, replace the default name "Config" with the new name before clicking **OK**:



20. Click **OK** to validate and close the Generate VR Configuration dialog box.


The XML file is generated and saved in the directory you specified. A warning message informs you that you need to restart your session to take this modification into account.

21. Exit your Version 5 session.
22. Restart a Version 5 session which is now correctly configured.
23. Key in the following command in the power input field:

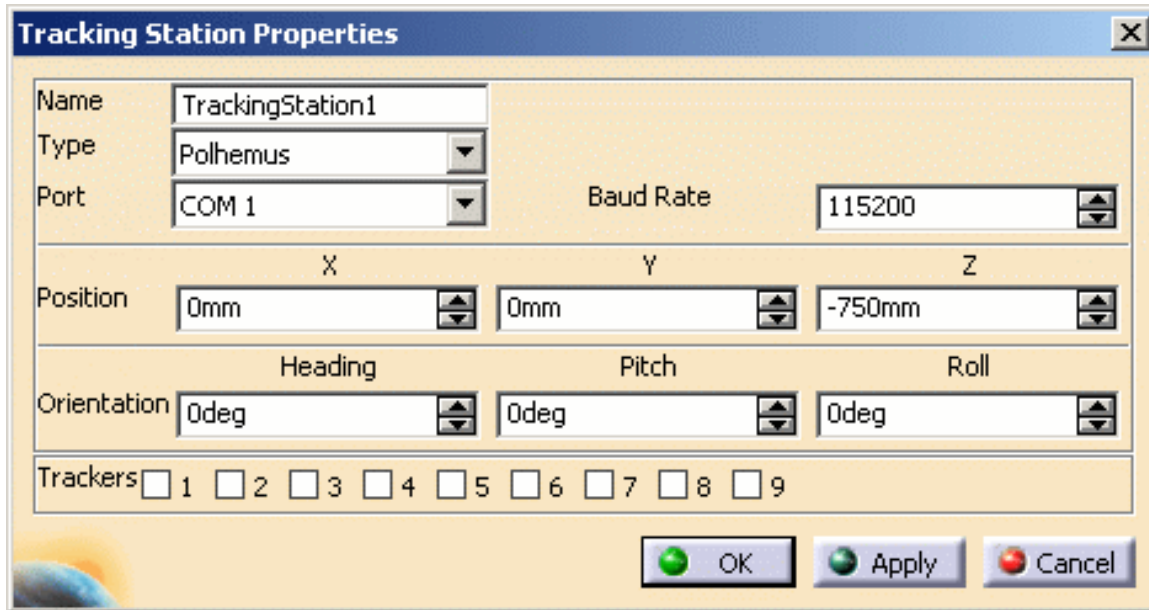
c:mpconfig

You are now ready to start!

If you want to use trackers, here are the additional steps you need to follow:

24. Click the **Create New Tracking Station**  icon to create the tracking station your trackers will be associated to and to define the type of tracker to be used.

The tracking Station Properties dialog box opens:



The image shows a screenshot of the 'Tracking Station Properties' dialog box. It has a title bar with a close button (X). The dialog is divided into several sections:

- Name:** A text field containing 'TrackingStation1'.
- Type:** A dropdown menu showing 'Polhemus'.
- Port:** A dropdown menu showing 'COM 1'.
- Baud Rate:** A numeric spinner field showing '115200'.
- Position:** Three numeric spinner fields for X, Y, and Z. X is '0mm', Y is '0mm', and Z is '-750mm'.
- Orientation:** Three numeric spinner fields for Heading, Pitch, and Roll. All are set to '0deg'.
- Trackers:** A row of nine checkboxes labeled 1 through 9, all of which are currently unchecked.
- Buttons:** At the bottom right, there are three buttons: 'OK' (with a green circle icon), 'Apply' (with a green circle icon), and 'Cancel' (with a red circle icon).

25. Type a *Name* (or leave the default) then specify the *Type* of the driver you intend to use (either Polhemus or IS900), serial communication *Port* number the driver is connected to as well as the communication *Baud Rate* on the serial port.

You can now choose a Flock of Birds or ART driver in the Type list.

Note: in case you choose a Version 5 compatible driver of your own (i.e. a "Custom" type), a new field will appear to the right asking you for the corresponding daemon name.

26. Indicate the *Position* of the tracking station.

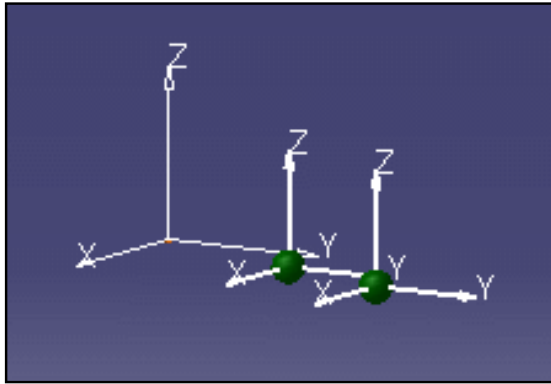
27. Indicate the tracking station:

- orientation in the *Heading* field
- vertical rotation in the *Pitch* field
- horizontal rotation in the *Roll* field.

- 28. In the *Trackers* area, identify the trackers to be connected to the tracking station by checking the corresponding figure.

Once you validated the tracking station definition, the tracking station and the trackers it is associated to are displayed in the geometry area.

In our example, we created a tracking station with two trackers:

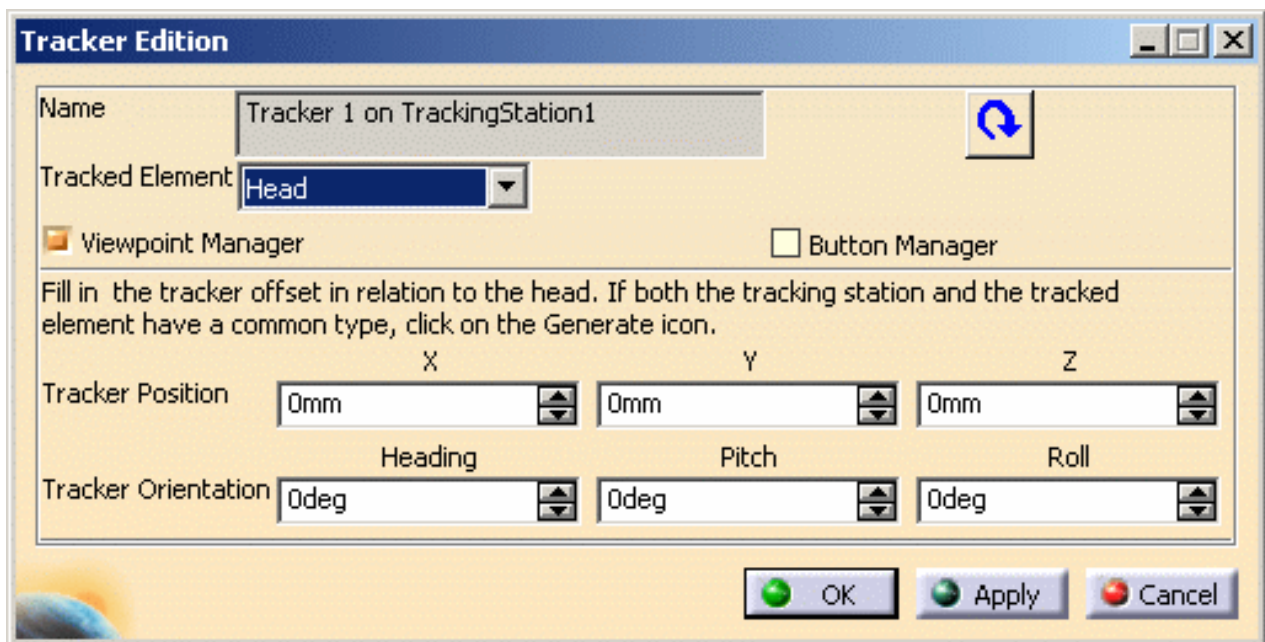


- 29. Double-click the head in the geometry area or the Head item in the specification tree to open the Head edition dialog box then specify the head type.

The next step is to edit the tracker. To do so:

- 30. Double-click the tracker symbol in the geometry area or the number of the tracker you wish to edit in the specification tree.

This opens the Tracker Edition dialog box:



- 31.** Select the *Tracked Element* from the pulldown list. The tracker may be attached to the head, the hand or another device.

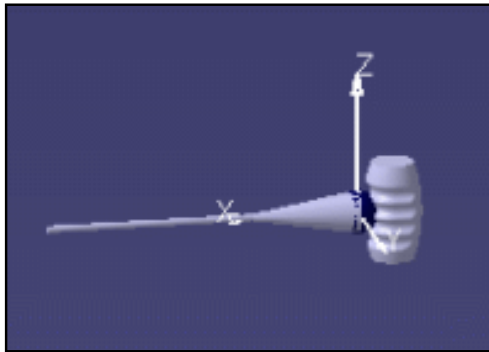


When several trackers are associated to the tracked element, activate the *Viewpoint Manager* option to indicate which tracker will be used.
For hand tracking, this option is named *Picking Manager*.

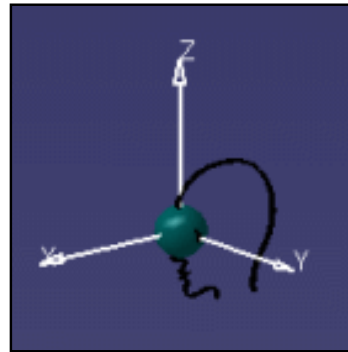


A tracker may be set as "Button Manager" when this tracker can also be used as a button device (for instance the IS900 or the ART tracker). Checking this option means that the button device will be used by the *VRCursor* command.

When the tracked element is selected, the tracker is attached to the corresponding element symbol in the geometry area and the color of the sphere changes according to the type of tracking:




Hand tracking



Head tracking




- 32.** If you are tracking hand or head, click the  button to automatically generate the matrices and set the standard, i.e. right, position for the tracker depending on the type of tracking station and tracked element.

The tracker symbol may be grayed and dotted if the matrices can be automatically calculated and if the information you entered does not match those we recommend.

If you want to define by yourself the tracker position, jump to the next step.


- 33. If you choose a "custom" tracker, the position and orientation data define the virtual tracker viewed by Version 5. These data correspond to the offset of the virtual tracker in relation to the physical tracker.

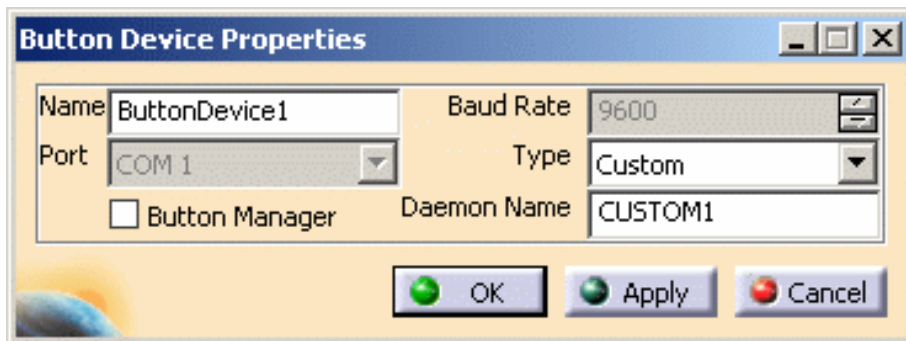


The  button is activated to let you define the tracker position in the device as well as its orientation:

- the *Heading* field lets you define the tracker orientation
- the *Pitch* field lets you define the vertical rotation
- the *Roll* field lets you define the horizontal rotation.




- 34. Click the **New Button Device**  icon to access the Button Device Properties dialog box:



You have to specify the following information:

- enter the name of the button device in the Name field
- select the device type in the Type pulldown list: CUSTOM, Mike, NeoWand or SpaceStick
- indicate the Dameon name.
- Button Manager: check this option if you want the button device to be used by the **VRCursor** command.
This capability is relevant for hand button devices onto which a tracker has been fixed, such as the Fakespace Neowand, the Virtual Presence SpaceStick, the ICIDO Mike, etc. In that case, you need to define a button device independently from the tracker.




- 35. Generate the configuration files by clicking the **Generate Configuration Files**  icon. For detailed information, refer to step 19 and repeat steps 19 to 22.

36. The next step depends on the driver and broker you are using:

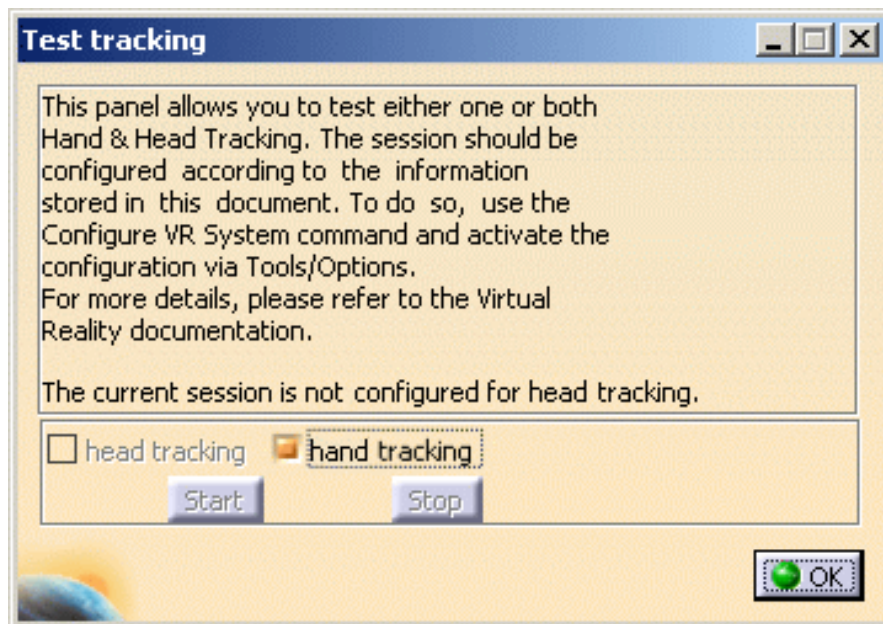
- if you are using a non-threaded broker and driver, you need to run them manually using the generated XML configuration file. The broker must be run first then, the driver. For detailed information on the commands to run, refer to [Head Tracking](#) or [Hand Tracking](#) according to the type of tracking you want to perform.

Then, you can restart a Version 5 session

- if you are using a threaded broker and driver, you can directly restart a Version 5 session: the driver is automatically run with the appropriate configuration.

37. In your Version 5 session, click the **Test viewpoint and VR cursor tracking**  icon to test either hand or head tracking, or even both if needed.

The Test tracking window appears and indicates for which tracking the current session is configured:



38. Specify the type of tracking you want to test (if your session is configured for both head and hand tracking) then click the **Start** button to validate.

As you use your tracking device, the tracker representation will move accordingly on screen.



Using Head and Hand Tracking Devices in Version 5



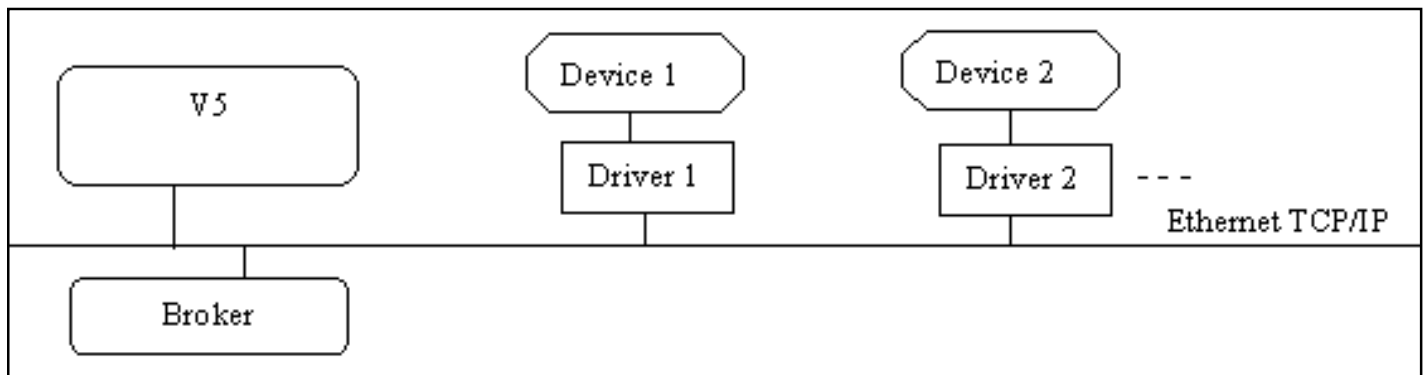
This task aims at giving you a general overview of the necessary virtual reality devices as well as two application examples for head tracking and hand tracking.

All the drivers supported by Version 5 are now threaded. You can still run them as standalone processes (as explained further in this section) but you can now activate them using the [Immersive System Assistant workbench](#) which enables you to generate a configuration file containing all the necessary data. These data will then be used to run the broker and the driver automatically when starting Version 5.

- [Device Overview](#)
- [Head Tracking](#)
- [Hand Tracking](#)

Device Overview

Using efficiently Virtual Reality services implies that you fully understand the Virtual Reality architecture. To do so, let`s have a quick look at the following scheme:



The virtual reality devices which can be handled are connected to Version 5 through a socket communication mechanism involving three actors: Version 5, a broker and a device driver.

Version 5 requires a broker to be able to use device drivers. A broker is a small application holding all the running driver communication port. This broker application always runs on the same machine as Version 5, whereas the drivers can run on another machine or another operating system.

Each driver declares itself to the broker thus implying that the broker should always be running before running a driver.

Broker and driver applications can be standalone processes or run within the Version 5 process (they are threaded processes). Some drivers may be threaded or not, depending on the devices. As a consequence, you can use either a threaded or standalone driver with a standalone broker.

To run a standalone broker or driver, you need to execute it explicitly, whereas running a threaded broker or driver only requires they be activated via the [Devices](#) tab in the *Tools->Options->Devices and Virtual Reality*.

Prior to using head or hand tracking in Version 5, you need to set up these actors proceeding the following way:



- 1. Start the broker.**

The broker role is to list and store any device declaration and to provide this list to Version 5 when requested. The broker must be started on the same host as the one of Version 5 session.

- 2. Start the driver.**

The driver will declare itself to the instantiated broker and will answer Version 5 when an event enumeration or an event sending is requested.

The driver may be launched on a different host as the one of the Version 5 session.

The driver sends events to the Version 5 application. Here are the four event types used:

- POSITION_EVENT for the tracker position and orientation in space
- VIEWPORT_EVENT for the coordinates of the screen corners as well as the left and right eye position
- ButtonPress which for the press button number
- ButtonRelease for the released button number.

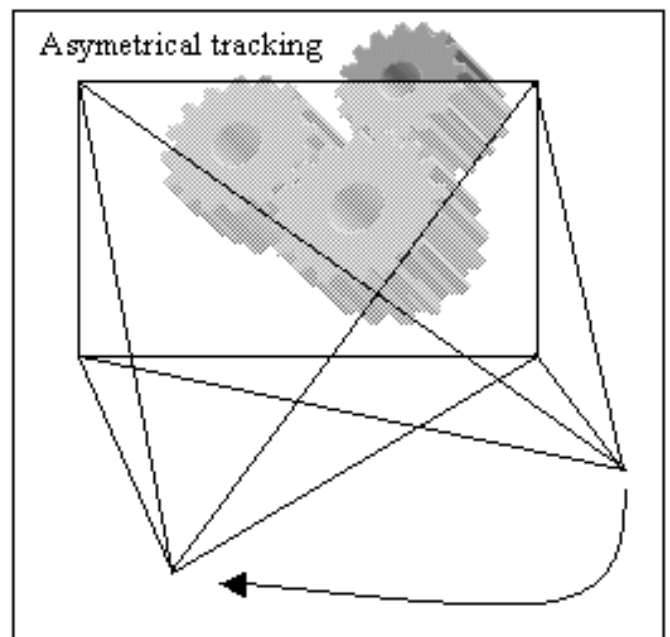
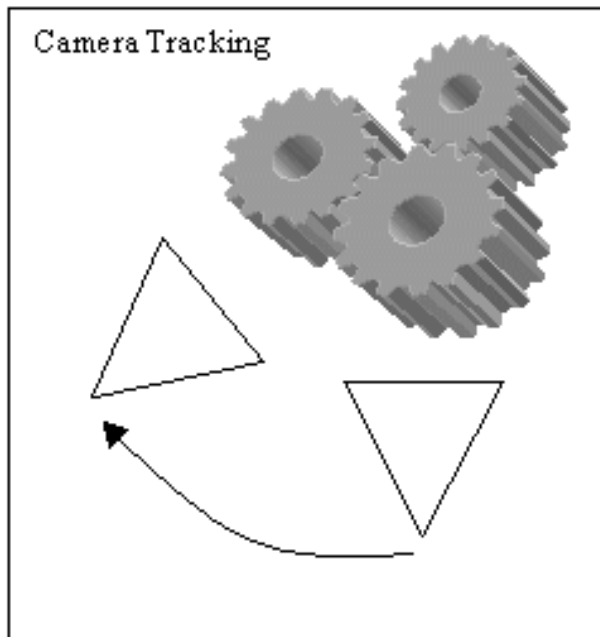
3. Start a Version 5 session.

When started, Version 5 will ask the broker for a list of devices and will ask each device for an event enumeration. In the meantime, these devices will be asked to start or stop event sending.

Head tracking

There are two kinds of head tracking:

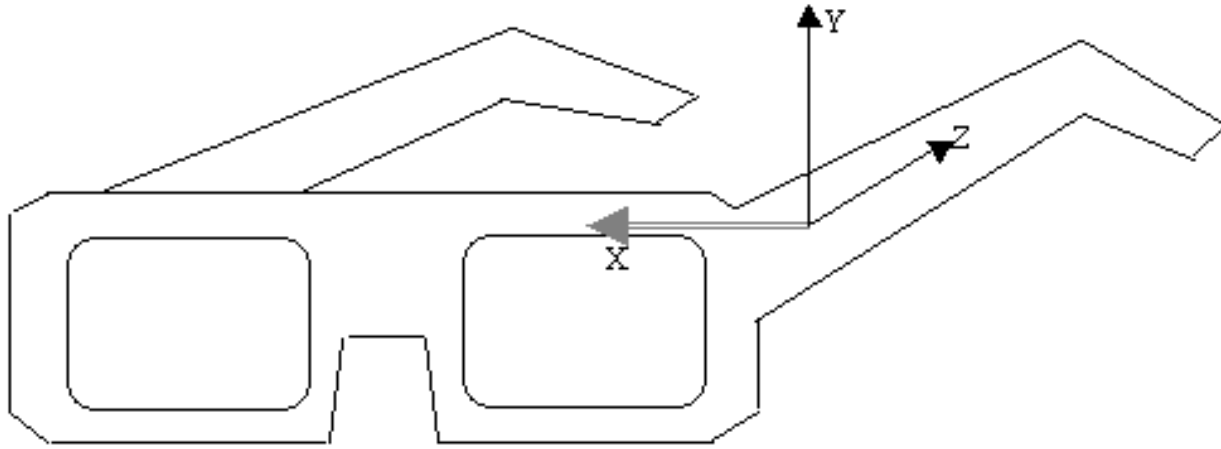
- *camera tracking* in which the tracking acts as if a camera moves in the model. The displayed image is the one filmed by the camera. In that case, the rendering projection pyramid keeps its original shape. This tracking is typically used for head mounted display or to simulate a camera
- *asymmetrical tracking* in which an image is computed and displayed on a screen that should be considered as being fixed in relation to the model, although you move in relation to the screen. This tracking is used for displaying images on fix screens and making the user feels as if the model were steady in the room as he moves in it.



The camera tracking functionality requires a `POSITION_EVENT` or a symmetric `VIEWPORT_EVENT` (head mounted display type).

The asymmetrical tracking functionality requires a `POSITION_EVENT` if the SGI multipipe management system is running, otherwise a `VIEWPORT_EVENT`.

When using a `POSITION_EVENT`, the head tracking requires the `POSITION_EVENT X axis` to be in the same direction as the vector defined by (Left Eye, Right Eye):



If the tracker is not fixed on the stereo glasses so that X is well aligned, you will need to ask the driver to apply an offset matrix.

If several matching events are available, a window will prompt you to choose an event when launching the VRViewTracking command: just select the event corresponding to the tracker dedicated to head tracking. You can also identify the convenient event thanks to its type and emitter driver name.



The following examples detail the viewpoint tracking installation procedures using a Polhemus Fastrak sensor and an IRIX platform.

The Version 5 driver managing the Polhemus Fastrak is a standalone (i.e. non threaded) driver named "PolExecDaemon".



1. The first step is to edit a configuration file to adapt it to your own configuration.

The configuration file holds necessary information for the driver: the communication baud rate on the serial port, the serial port number, the events to send, the matrix you could use to virtually offset the trackers from their mounting point, the screen definition and the eye position for the VIEWPORT_EVENT.

Two examples of configuration files are detailed in this task but bear in mind that there are no specific settings for using the Polhemus Fastrak. The dip switch on the Polhemus box should correspond to what is declared in the configuration file.

For your information, the settings used in Dassault Systèmes are:

Select (number of devices): Down Up Up Up (one sensor only)

I/O Select: Up Up Up Down Down Up (serial port 115200 bps).

Be careful to use always the same hemisphere during viewpoint tracking. Refer to the Polhemus documentation for detailed information.

The Immersive Assistant workbench enables you to generate automatically the necessary configuration file for head tracking. For detailed information, refer to [Working With the Immersive System Assistant](#) in this guide.

However, you can still edit the configuration file manually, if desired. To do so, follow the instructions detailed below.

Example 1

Let's describe a configuration file assuming you use a Polhemus Fastrak for tracking both head and hand. Your display system is a bench or a multi-channel display and you use the `mpconfig` functionality.

In this case, you need the driver to send two POSITION_EVENTS.

The # symbol indicates a comment in the configuration file and the opposite text explains each of the following instructions:

```
SERIALPORT      "COM1"
BPS              38400
TRACKER 0 POSITION_EVENT
TRACKER 1 POSITION_EVENT
```

Specify the serial communication port number. You can use either COM1, COM2 or COM3.

Specify the communication baud rate with the Polhemus Fastrak. Usual baud rates are 38400 and 115200.

#Intersense Head Tracker

```
MATRIX 0
[0,1,0]
[0,0,-1]
[-1,0,0]
[0,0,0]
```

Here you can type as many lines as events you need. Specify the tracker number: a Polhemus Fastrak has up to four trackers plugged in. Specify the desired event for each tracker. You can choose among four types of events by selecting them using the keywords POSITION_EVENTS, VIEWPORT_EVENT, BUTTON_PRESS and BUTTON_RELEASE. You can ask for different events on the same tracker number.

#Intersense Wand

```
MATRIX 1
[0,1,0]
[0,0,-1]
[-1,0,0]
[0,0,0]
```

Specify the matrix to be applied to the POSITION_EVENT coming from tracker 0. This matrix means that the drivers will send to the application the Y-tracker axis as X axis, the Z-tracker axis as Y axis and the X-tracker axis as Z axis.

```
SCREEN
{
  UPPERLEFT [-700, 300, -430]
  UPPERRIGHT [-700, 300, 430]
  LOWERRIGHT [-700, 900, 430]
}
```

If you do not apply any matrix, use an identity matrix.

Keep a screen section (detailed hereafter) even though it is not used.

Now, suppose you want to use the Polhemus Fastrak to track both hand and head without using the **mpconfig** functionality. In this case, you will need asymmetrical tracking just as you have done it before but, as you will not use mpconfig, you will need the driver to send a POSITION_EVENT for the hand and a VIEWPORT_EVENT for the head.

The configuration file will then look like this:

```
SERIALPORT "COM1"
BPS      115200
TRACKER 0 VIEWPORT_EVENT
TRACKER 1 POSITION_EVENT
MATRIX 1
[0,0,1]
[1,0,0]
[0,1,0]
[0,0,0]
```

```
SCREEN
{
  UPPERLEFT [-700, 300, -430]
  UPPERRIGHT [-700, 300, 430]
  LOWERRIGHT [-700, 900, 430]
}
```

```
LEFTEYE [0.0, 0.0, 61.0]
RIGHTEYE [0.0, 0.0, 124.0]
```

For the VIEWPORT_EVENT, you need to specify the corner coordinates of the projection screens. These coordinates are to be expressed within the same reference frame (R0) as the data sent by the tracker.

The screen is assumed to be rectangular.

You also need to specify the eye position in the mobile tracker reference frame. We call "mobile tracker reference frame" the reference frame defined by the x, y, z, h, p and r coordinates sent by the tracker. Express these coordinates in millimeters.



Here are some additional information on the above-mentioned parameters:

- **MATRIX**

Polhemus will send an event to PolExecDaemon driver which has been defined in the axis system of the transmitter. In turn, PolExecDaemon will send an event to Version 5 but this event will be modified using the matrix you applied. Generally, the matrix is used for sending POSITION_EVENT thus, in case you are only using VIEWPORT_EVENT, the matrix parameter is not mandatory

- **SCREEN**

This parameter defines the coordinates of the visualization screen in the Polhemus axis system. The screen coordinates have to be measured manually. The result depends on the the position and orientation of the Polhemus transmitter.

For instance, suppose you are using a transmitter positioned in front of the screen with X pointing out the screen, Y down and Z right (cf. the Axis system engraved on the Polhemus transmitter box). In this case, you get coordinates where X is constant and negative and the other values vary along the YZ plane. This corresponds to the values detailed in the above example:

```
SCREEN
{
  UPPERLEFT [-700, 300, -430]
  UPPERRIGHT [-700, 300, 430]
  LOWERRIGHT [-700, 900, 430]
```

}

- **LEFTEYE** and **RIGHTEYE**

These parameters let you define the position of the eyes in the mobile tracker axis system. The result depends on how the Polhemus sensor is attached to the user's head.

For instance, you can attach the Polhemus sensor to the left branch of the stereo glasses which means that X points to the screen, Y is up and Z goes from left eye to right eye. This axis system corresponds to the default as explained in the Polhemus manual (refer to the Polhemus Fastrak User's Manual Revision F for detailed information). In that case, the eyes are aligned exactly on the Z axis of the sensor and the coordinates you get correspond to those detailed in the above example:

LEFTEYE [0.0, 0.0, 61.0]

RIGHTEYE [0.0, 0.0, 124.0]

These coordinates have to be measured manually.

Example 2

Let's take a look at a second example and consider the following case: you want to use the Polhemus Fastrak to track both hand and head and your display system is a Head Mounted Display.

You will thus need a camera type tracking. You also need the driver to send a POSITION_EVENT for the hand and either a POSITION_EVENT or a VIEWPORT_EVENT for the head. If the driver sends a POSITION_EVENT for the head, no stereo viewing is available and you need to adjust the viewpoint angle for your head mounted display in Version 5. To do so, create a named view by selecting the **View->Named Views...** command then edit its properties.

Note: you cannot adjust its ratio in case the head mounted display optics are not symmetric.



This camera is saved in the .CATProduct file under the application node. The POSITION_EVENT must have its X axis going to the right, its Y axis up and its Z axis backwards. Otherwise (as we explained it before), you have to ask the driver to apply a rotation matrix. You can use the first configuration file for this system.

In case you use a VIEWPORT_EVENT, the configuration file should be as follows:

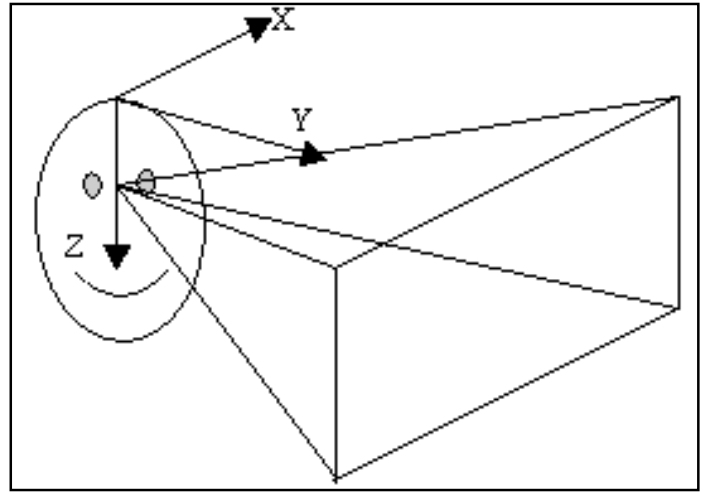
```
SERIALPORT      "COM1"
BPS             115200
TRACKER 0 VIEWPORT_EVENT
TRACKER 1 POSITION_EVENT
MATRIX 1
[0,0,1]
[1,0,0]
[0,1,0]
[0,0,0]
```

Use this section to describe the HMD fields of view. Describe the desired virtual projection pyramid.

All the figures are to be expressed in the mobile tracker (fixed on the head) reference frame centered between right and left eye. For instance, if the tracker is mounted such as Z is going down and Y forward, the coordinates are as shown below:

```
HMDSCREEN
{
  UPPERLEFT [700, -900, -720]
  UPPERRIGHT [700, 900, -720]
  LOWERRIGHT [700, 900, 720]
}
```

LEFTEYE [0.0, 0.0, 61.0]
RIGHTEYE [0.0, 0.0, 124.0]



Here, the pixel ratio is 1.25.



We recommend you to pay attention to the coordinates you enter. The more careful you are, the better the results will be.

2. Run the Virtual Reality broker using the following command:

```
./.../B13/irix_a/code/command/catstart -run CATVisVRBroker
```

where *./.../* is the Version 5 installation path. For a standard installation, the path should be */usr/DassaultSystemes*.

3. Run the tracker device driver for viewpoint tracking:

```
./.../B13/irix_a/code/command/catstart -run PolExecDaemon -object machine_name  
configuration_file
```

where *configuration_file* is the full path name of the configuration you edited in step 1 and *machine_name* is the name of the machine.



This driver may be used with any device compliant with the Fastrak protocol.

4. Start Version 5:

```
./.../B13/irix_a/code/command/catstart
```

If you want to use stereoscopic viewing, you have to perform Steps 5 to 12 otherwise, jump to Step 13.

5. Select the **Tools->Options...** command then the Visualization tab from the **General->Display** category.
6. In the "Stereo enable" area, check the **On** option.
7. Click **OK** to validate your parameters.
8. Restart Version 5.
9. In the power input field, enter the following command:

```
c:stereoscopic
```

10. Click **OK**.

11. In the power input field, enter:

```
c:VRViewTracking
```

The Viewpoint Tracking is activated. It can be simply deactivated by re-entering the `c:VRViewTracking` command in the power input field

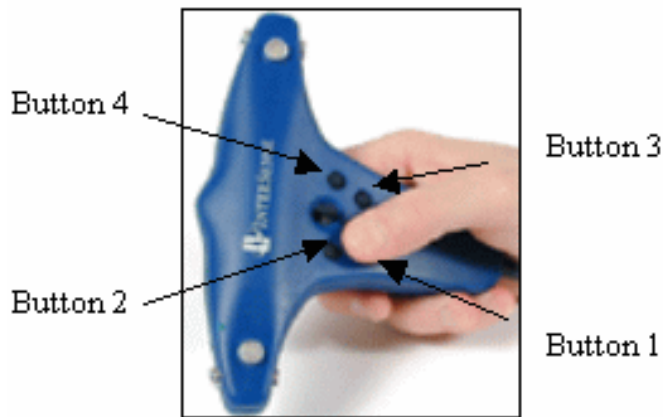
Before using this command, make sure that a driver is running and able to send `POSITION_EVENT` or `VIEWPORT_EVENT`.

Intersense

Since the Intersense firmware protocol is an enriched Fastrak protocol, the PolExecDaemon Version 5 driver can communicate with an Intersense IS900 tracker.

The above-mentioned instructions are still valid in this case.

In addition, the Intersense IS900 supports a Wand hand tracker. This specific hand tracker has four buttons the driver is able to decode as `ButtonPress` and `ButtonRelease` events.



To configure the driver with these two events, use the keywords `BUTTON_PRESS` and `BUTTON_RELEASE` as explained above.

Use the following matrix in the `PolExecDaemon` driver so that the attached reference frame is as required by the Version hand tracking functionality:

```
MATRIX 1
[0, 1, 0]
[0, 0, -1]
[-1, 0, 0]
[0, 0, 0]
```

Ascension Flock of Birds



The Ascension Flock of Birds tracking system driver is now supported for head tracking on Windows and UNIX. Note that from V5R13 onwards, you do not need to install the `Bird.dll` library anymore.

This driver allows to measure in real time the position and orientation of several bodies in space. It can be used as a standalone executable (called "`CATFlockOfBirdsDriver`") or as a threaded driver.

To use this driver as a standalone process, follow the instructions below:

On Windows

Go to the following installation directory

```
\install_root\intel_a\code\command
```

and enter the command:

```
catstart -run CATFlockOfBirdsDriver [-f Frequency] [-com SerialPortNumber] [-bps
SerialPortSpeed] [-host BrokerHostName] [-m TransformationMatrix] [-n NumberofDevices] [-
mouse DeviceNumber]
```

On UNIX

Go to the following installation directory

```
/install_root/OS_a/code/command
```

where "OS_a" is:

- aix_a
- hpux_b
- irix_a
- solaris_a.

and enter the command:

```
./catstart -run CATFlockOfBirdsDriver [-f Frequency] [-com SerialPortNumber] [-bps  
SerialPortSpeed] [-host BrokerHostName] [-m TransformationMatrix] [-n NumberofDevices] [-  
mouse DeviceNumber]
```

Below is a description of the arguments:

-f	Event sending frequency from the CATFlockOfBirdsDriver to the Version 5 session. If not specified, a default value of 100 Hz is used
-com	Serial port number to which the Flock of Birds is connected. If not specified, a default value of 1 is used, meaning "COM1" on Windows systems.
-bps	Serial port speed. If not specified, the driver tries to connect to the Flock of Birds at the speed of 115200 bps.
-host	Name of the host on which the CATVisVRBroker is running. If not specified, LocalHost is assumed.
-m	Position and orientation received from the Flock of Birds are multiplied by this matrix before being sent to the Version 5 session. The matrix should be described in lines using the following format: -m [m11,m12,m13][m21,m22,m23][m31,m32,m33] If not specified, the Identity matrix is assumed.
-n	Number of Flock of Birds running at the same time. If not specified, one Flock of Birds is assumed.
-mouse	Device number of the Flock of Birds to which an Ascension 6D Mouse is connected. If not specified, no 6D Mouse is supposed to be used. The numbering of the devices start with number 1.
-debug	Activates the display of error messages.

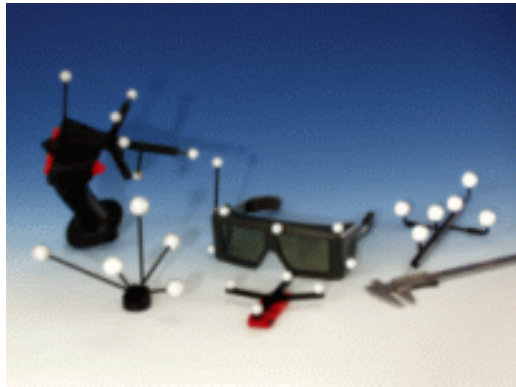
However, bear in mind that this driver sends POSITION_EVENT only.

This driver is also supported for [hand tracking](#).



ART Optical tracking system

Version 5 now supports the ART driver. This driver can be used as a standalone executable (called "CATARTDriver") or as a threaded driver.



How does it work?

The ART Optical tracking system sends position and button information at a specific TCP/IP address on a specified port number. The Version 5 driver reads this ethernet port and decodes the information before translating it. The translated information is then sent by the driver to the Version 5 session in a Version 5 format.

To use this driver as a standalone process, enter the following command:

On Windows

```
/installation_path/intel_a/code/command/catstart -run CATARTDriver -h
```

The "-h" option will display help information.

On UNIX

```
/install_path/OS_a/code/command/catstart -run CATARTDriver -h
```

where "OS_a" is:

- `aix_a`
- `hpux_b`
- `irix_a`
- `solaris_a.`

Note that this driver sends:

- POSITION_EVENT
- BUTTON_EVENT
- SLIDER1_EVENT
- SLIDER2_EVENT.

This driver is also supported for [hand tracking](#).



Hand tracking

The following two examples are designed for hand tracking using:

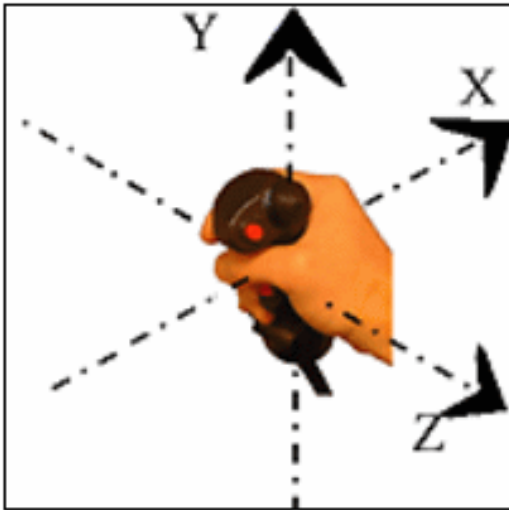
- a Joystick Spacestick VirtualPresence with a Polhemus sensor and the DMU Navigator 3 product
- a FakeSpace Neowand.

Note that the Immersive Assistant workbench enables you to generate automatically the necessary configuration file for hand tracking. For detailed information, refer to [Working With the Immersive System Assistant](#) in this guide.

However, you can still edit this file manually if desired. To do so, follow the instructions detailed below.

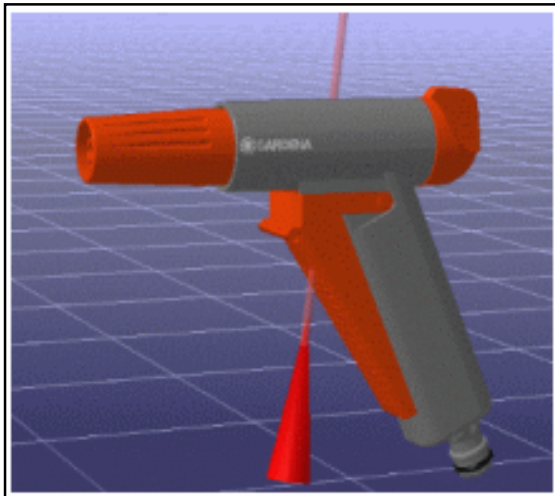
The **VR Cursor** command switches on and off the hand tracking functionality with the following prerequisite: one or more drivers should be running and able to send a POSITION_EVENT, a ButtonPress and a ButtonRelease. The command will look for the first driver able to send POSITION_EVENT then, ButtonPress and ButtonEvent.

The POSITION_EVENT used for hand tracking should have its X axis pointing to the right, its Y axis up and its Z axis backward as shown below:



If the tracker is not fixed this way in the hand device, you will have to ask the driver to apply an offset matrix.

When "On", hand tracking displays a red laser pointer. This pointer is collocated with the hand tracker so that you really feels the pointer getting out of your hand. This feeling is reinforced in stereoscopic display:



The buttons enable you to pick within the model and to manipulate the viewpoint. There are two manipulation modes: **Examine mode** and **Fly mode**. You can switch between the two simply by running the **VRFly** command. The Examine mode is the default mode when running the **VRCursor** command.

You can also move the model objects by snapping the 3D compass to them. The 3D compass will then follow your hand moves.

Note: setting the compass to "Snap automatically to selected object" lets you snap the compass to any objects you can pick with the laser pointer.

Now let's go to using hand tracking!

Joystick Spacestick



1. Run the PolExecDaemon.

2. Run the Joystick3DExecDaemon using the following command:

```
./.../B13/{OS_a}/code/command/catstart -run "Joystick3DExecDaemon brokerHost serialPort"
```

where

- ./.../ is the Version 5 installation path. For a standard installation, the path should be /usr/DassaultSystemes/
- {OS_a} is the name of your operating system (irix_a, etc.)
- brokerHost is the name of the machine on which the device broker and Version 5 are running
- serialPort is the port to which the joystick is connected (e.g.: COM1, COM2, COM3 or COM4).

There is no configuration file for this driver.



Another method to run the driver is to first create an icon on your desktop, then drag the Joystick3DExecDaemon.exe file from the explorer and drop it onto the desktop (this file can be found in the installation directory, in intel_a\code\bin).

Edit the icon properties and modify the target file to

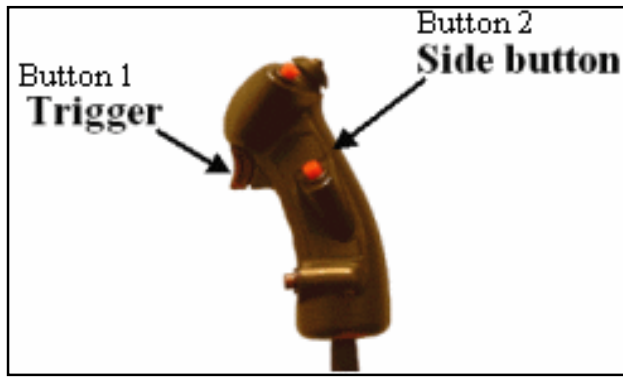
```
c:\...\intel_a\code\bin\Joystick3DExecDaemon.exe brokerHost serialPort.
```

Then, you just have to double-click the icon to run the driver.

3. For the Polhemus version (a Polhemus tracker is fixed inside the joystick), use the following matrix in the PolExecDaemon driver so that the attached reference frame matched the Version 5 hand tracking functionality requirements:

```
MATRIX  
[0,0,1]  
[1,0,0]  
[0,1,0]  
[0,0,0]
```

The buttons are located as described on the picture below:



4. Run the DMU Immersive Review configuration then the DMU Navigator 3 product.

Fakespace Neowand

The Version 5 driver managing the Fakespace Neowand buttons is a standalone (i.e. non threaded) driver named "CATNeoWandDriver".



1. Run the CATNeoWandDriver using the following command:.

```
./.../{OS_a}/code/command/catstart -run CATNeoWandDriver brokerHost serialPort"
```

where

- /.../ is the Version 5 installation path. For a standard installation, the path should be /usr/DassaultSystemes/Bn ("n" representing the release number)
- {OS_a} is the name of your operating system (irix_a, etc.)
- brokerHost is the name of the machine on which the device broker and Version 5 are running
- serialPort is the port to which the joystick is connected (e.g.: COM1, COM2, COM3 or COM4).

There is no configuration file for this driver.



Another method to run the driver is to first create an icon on your desktop, then drag the CATNeoWandDriver.exe file from the explorer and drop it onto the desktop (this file can be found in the installation directory, in intel_a\code\bin).

Edit the icon properties and modify the target file to

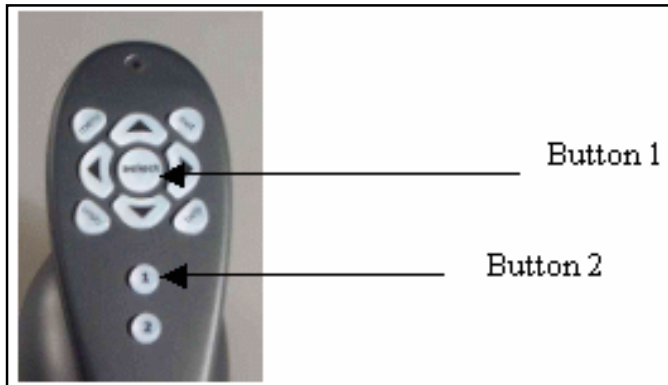
```
c:\...\intel_a\code\bin\CATNeoWandDriver.exe BrokerHostName serialPort.
```

Then, you just have to double-click the icon to run the driver.

2. In case a Polhemus tracker is fixed inside the NeoWand, use the following matrix in the PolExecDaemon driver so that the attached reference frame matches the Version 5 hand tracking functionality requirements:

MATRIX
[0, 1, 0]
[0, 0, -1]
[-1, 0, 0]
[0, 0, 0]

The buttons are located as shown in the picture below:



Click the thumbnail  to see the full-size picture.

Icido Thready Motion Controller

In addition to the Fakespace Neowand, Version 5 supports Icido Thready Motion Controller (also known as "Mike"). The driver managing this device is named "CATMikeDriver" and should be used the same way you do with CATNeoWandDriver detailed above.

Ascension Flock of Birds



The Ascension Flock of Birds tracking system driver is now supported for hand tracking on Windows and UNIX. Note that from V5R13 onwards, you do not need to install the Bird.dll library anymore. This driver allows to measure in real time the position and orientation of several bodies in space. It can be used as a standalone executable (called "CATFlockOfBirdsDriver") or as a threaded driver.

To use this driver as a standalone process, follow the instructions below:

On Windows

Go to the following installation directory

```
\install_root\intel_a\code\command
```

and enter the command:

```
catstart -run CATFlockOfBirdsDriver [-f Frequency] [-com SerialPortNumber] [-bps SerialPortSpeed] [-host BrokerHostName] [-m TransformationMatrix] [-n NumberofDevices] [-mouse DeviceNumber]
```

On UNIX

Go to the following installation directory

```
/install_root/OS_a/code/command
```

where "OS_a" is:

- aix_a
- hpux_b
- irix_a
- solaris_a.

and enter the command:

```
./catstart -run CATFlockOfBirdsDriver [-f Frequency] [-com SerialPortNumber] [-bps SerialPortSpeed] [-host BrokerHostName] [-m TransformationMatrix] [-n NumberofDevices] [-mouse DeviceNumber]
```

Below is a description of the arguments:

-f	Event sending frequency from the CATFlockOfBirdsDriver to the Version 5 session. If not specified, a default value of 100 Hz is used
-com	Serial port number to which the Flock of Birds is connected. If not specified, a default value of 1 is used, meaning "COM1" on Windows systems.
-bps	Serial port speed. If not specified, the driver tries to connect to the Flock of Birds at the speed of 115200 bps.

- host Name of the host on which the CATVisVRBroker is running.
If not specified, LocalHost is assumed.

- m Position and orientation received from the Flock of Birds are multiplied by this matrix before being sent to the Version 5 session. The matrix should be described in lines using the following format:
-m [m11,m12,m13][m21,m22,m23][m31,m32,m33]
If not specified, the Identity matrix is assumed.

- n Number of Flock of Birds running at the same time.
If not specified, one Flock of Birds is assumed.

- mouse Device number of the Flock of Birds to which an Ascension 6D Mouse is connected.
If not specified, no 6D Mouse is supposed to be used. The numbering of the devices start with number 1.

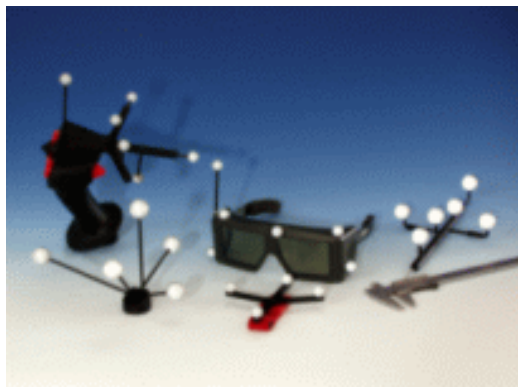
- debug Activates the display of error messages.

However, bear in mind that this driver sends POSITION_EVENT only.



ART Optical tracking system

Version 5 now supports the ART driver. This driver can be used as a standalone executable (called "CATARTDriver") or as a threaded driver.



How does it work?

The ART Optical tracking system sends position and button information at a specific TCP/IP address on a specified port number. The Version 5 driver reads this ethernet port and decodes the information before translating it. The translated information is then sent by the driver to the Version 5 session in a Version 5 format.

To use this driver as a standalone process, enter the following command:

On Windows

```
\installation_path\intel_a\code\command\catstart -run CATARTDriver -h
```

The "-h" option will display help information.

On UNIX

```
/install_path/OS_a/code/command/catstart -run CATARTDriver -h
```

where "OS_a" is:

- aix_a
- hpux_b
- irix_a
- solaris_a.

Note that this driver sends:

- POSITION_EVENT
- BUTTON_EVENT
- SLIDER1_EVENT
- SLIDER2_EVENT.

This driver is also supported for [head tracking](#).



Tracking calibration

In case you intend to use a multi-channel display system with head tracking, you should add some information related to the tracking system inside the MPK configuration file. These information should be added at the end of the file, before the last closing bracket "}".

They should all begin with a comment symbol "#". Even though there are commented out, they are used by Version 5. The only used information are those included in the "userdata" section. The measure unit for all the numbers is the millimeter.

As required by the head tracking functionality, the mobile tracker reference frame should have its X axis pointing right, its Y axis pointing up, and its Z axis pointing backward. However, it may not have its origin located exactly between the two eyes. The "eye-offset" keyword enables you to specify the middle eye point coordinates in this mobile tracker reference frame.

The fix tracker reference frame may be different from the MPK frame. Use the "tracker-X", "tracker-Y" and "tracker-origin" keywords to specify it. In the example below, the fix tracker frame has its X axis pointing backward., its Y axis pointing to the right and its origin directly 1524 mm under the MPK origin.

If you used the millimeter as unit in the MPK file, the keyword "MPK_unit_in_mm" should be valuated to 1. If you used meter for instance, it should be valuated to 1000.

```
# All dimensions are to be specified in mm.
# - eye-offset specifies the eye coordinates in the mobile tracker frame
# - IPD is the inter-pupillary distance
# - the tracker related data describe the fixed tracker frame in the MPK global frame
# - MPK_unit_in_mm defines the MPK data unit.
# userdata
# {
#   eye-offset   [0,-20,0]
#   IPD          57
#   tracker-X    [0,0,-1]
#   tracker-Y    [1,0,0]
#   tracker-origin [0,-1524,0]
#   MPK_unit_in_mm 1
```

A standard session example



Suppose the following case:

- you use a three-channel display system in a reality center
- you also use a three-pipes SGI machine. The display system has three projectors, one per channel, projecting active stereo
- you want your head and hand to be tracked in front of the screens
- your tracking system is an Intersense IS900
- there are two trackers, one for the head and one for the hand, both plugged on the IS900
- the hand tracker is an Intersense wand with four buttons.

You have already set up the system, i.e. you have all the necessary configuration files and your CATSettings are up-to-date.

Moreover, you have set the "Virtual reality starting mode" option to "Manual" in the Tools->Options->General->Devices tab since the Intersense driver is a standalone, i.e. non threaded, driver.

The scenario detailed hereafter will show you how to use the system.



1. Power up the Intersense IS900

2. Open a shell window then run the broker (this is a non threaded case)

3. Open another shell window then run the Version 5 IS900 driver.

Because of the configuration file you use, the driver will send two POSITION_EVENTS (one per tracker), a ButtonPress and a ButtonRelease

4. Open a third shell window then run a Version 5 session

5. Open a .CATProduct model file

6. Key in the following command in the power input field:

```
c:mpconfig
```

then click OK when a window appears to indicate that the Multipipe mode is started.

7. Key in the following command in the power input field:

```
c:VR View Tracking
```

then choose the head tracker POSITION_EVENT before clicking **OK**.

8. Key in the following command in the power input field:

```
c:VR Cursor
```

9. Key in the following command in the power input field:

```
c:VRFly
```

to select the convenient manipulation mode for your session

10. Select the **View->Full Screen** command.

The system is running.


You can customize the commands entered in the power input field so that they appear as push buttons in a toolbar. You can also assign a keyboard shortcut to them if you wish so.

When you want to open a new model or quit the session:

1. Deactivate the full screen mode by unchecking the Full Screen contextual command
2. Key in **c:VR View Tracking** in the power input field to stop head tracking
3. Key in **c:VR Cursor** in the power input field to stop hand tracking
4. Key in **c:mpconfig** in the power input field to stop the multipipe management
5. Select the **File->Open** or the **File->Exit** command.



Using the Joystick Control Panel

 This section provides information about using the Joystick Control Panel in order to set the joystick behavior.

 If you are running Windows 2000 or Windows XP, you must setup the joystick prior to run the joystick command. To do so, access the **Tools->Options->General->Devices and Virtual Reality->Devices** tab:

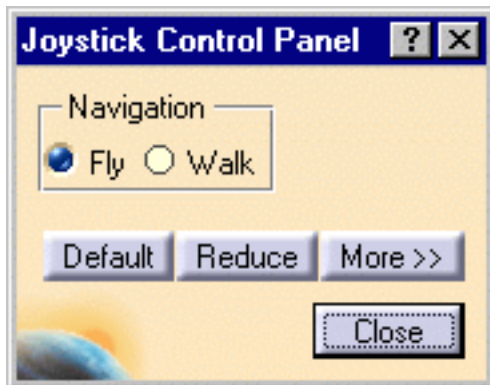
- activate the "Automatic" Virtual reality starting mode
- click the Joystick icon in the Automatically Started Daemons area.

Otherwise, you can still run the joystick daemon named CATJoystickDriver. Refer to the [Joystick Spacestick](#) part to read more on how to run a joystick daemon.

 **1.** In the power input field, key in the following command:

c:joystick

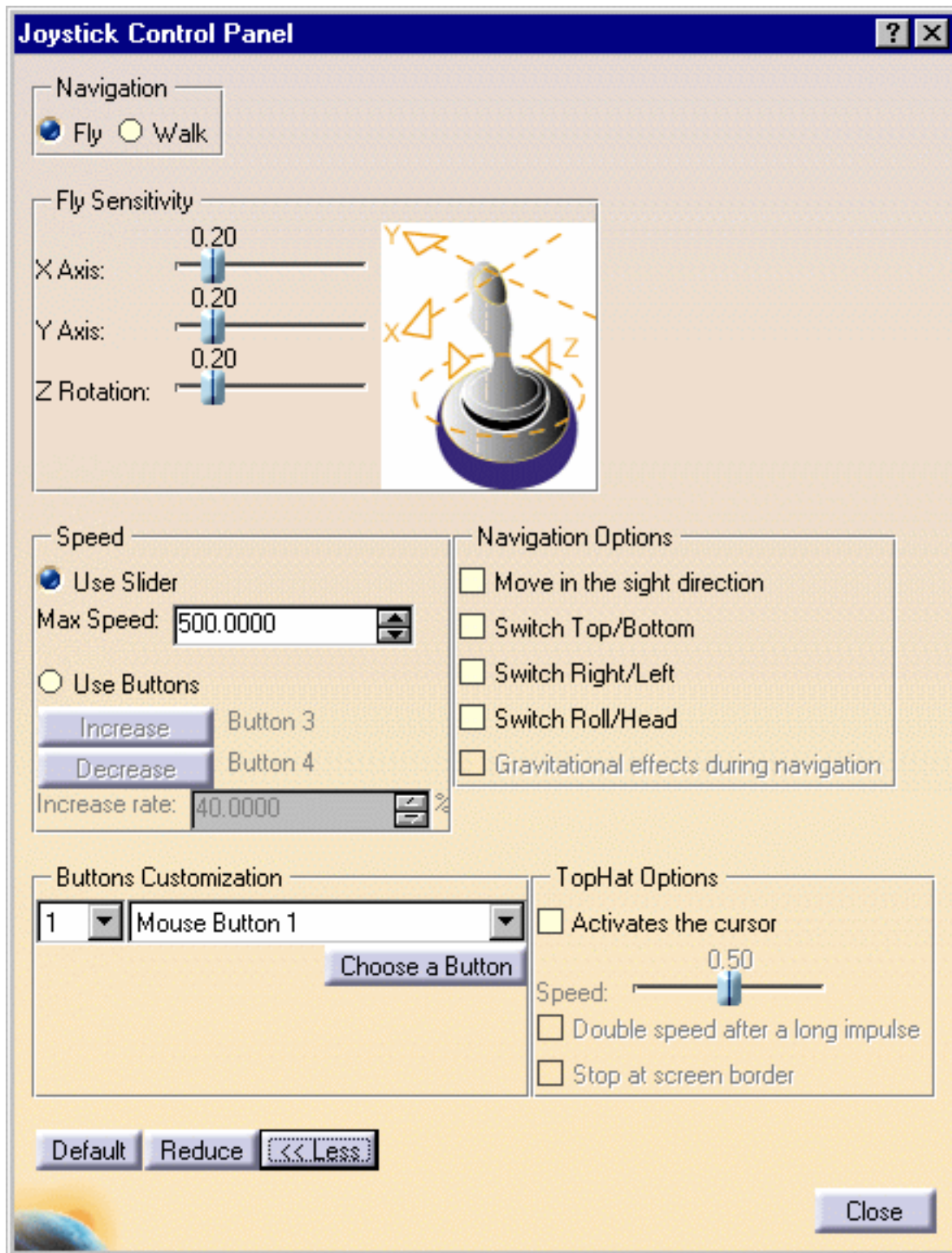
The Joystick Control Panel opens:



2. Select your Navigation mode by clicking the corresponding radio button: Fly or Walk.

Note: these navigation modes are totally independent from the standard Fly and Walk modes available in the View toolbar.

3. Click the **More >>** button to display detailed information about the joystick:



4. In the Fly Sensitivity area, use the sliders to define the sensibility when rotating around the X, Y and Z axes.

5. The Speed area lets you choose between the joystick slider or the joystick buttons to set the maximum displacement speed:
 - **Use Slider** option enables you to enter a number comprised between 0 to the maximum speed
 - **Use Buttons** option activates the **Increase** and **Decrease** buttons to let you set the buttons that will increase or decrease speed.
6. Still in the Speed area, enter the Increase rate by which the speed will be increased when pressing the joystick button.
7. The Button Customization area enables you to assign an action to the button of your choice. First select an action from the pulldown list then click **Choose a Button** and choose the desired button.
8. In the Navigation Options area, you can check the displayed checkboxes to:
 - move in the sight direction. This option is used in combination with head tracking
 - invert the joystick top and bottom
 - invert the joystick right and left
 - invert Y axis and Z rotation.




When checked, the "Gravitational effects during navigation" option indicates that gravitational effects are enabled while navigating in Walk Mode. This option must be activated (and also deactivated) via the **Tools->Options->General->Display->Navigation** tab.

Note that gravitational effects are not relevant for Fly mode navigation.

9. The TopHat options area lets you use the cursor. First click the "Activates the cursor" checkbox to activate the TopHat options.



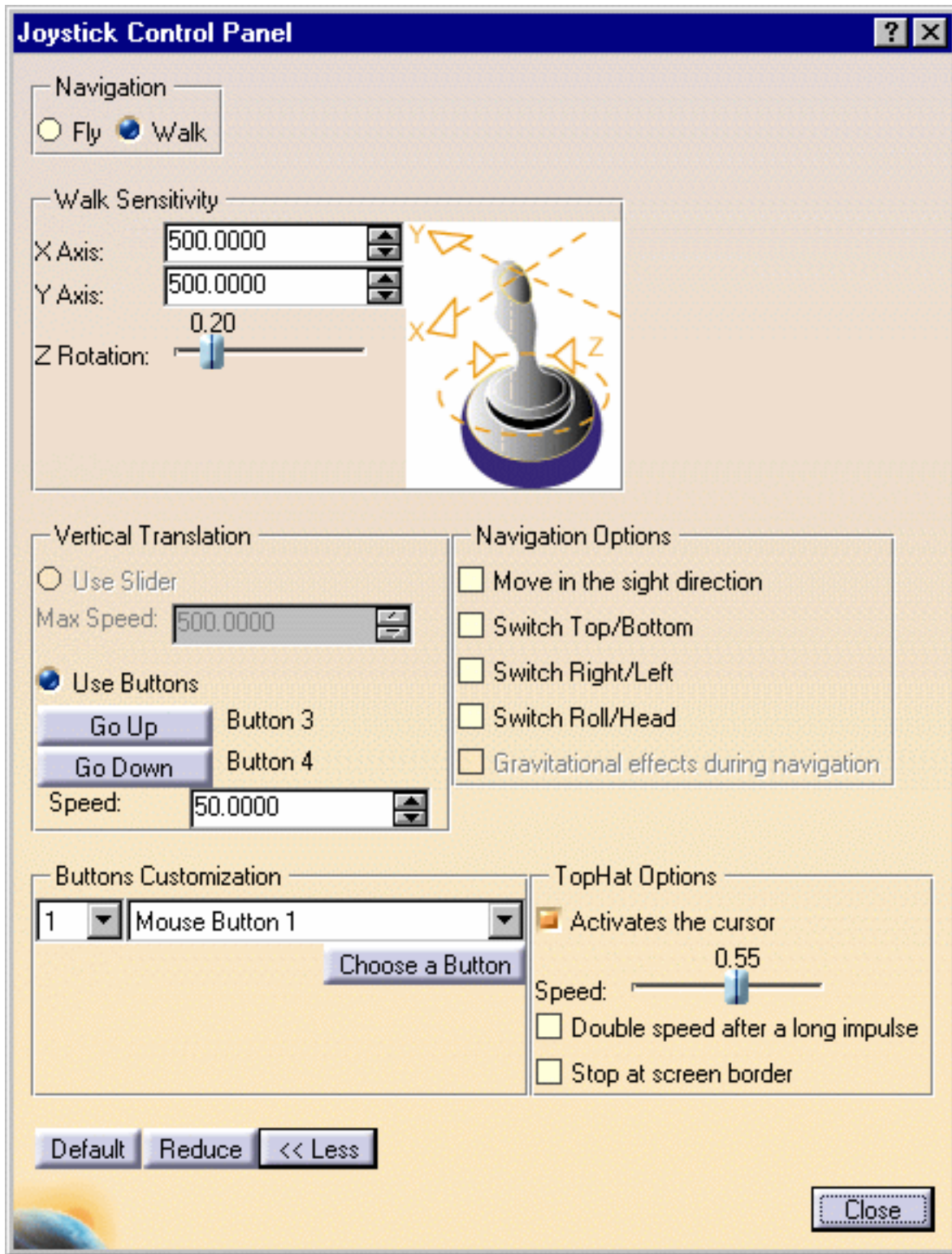
A cross  appears on the screen to simulate the mouse. You can then perform the same interactions as you would do with a mouse.

10. Use the Speed slider to set the cursor speed.
11. Check the "Double speed after a long impulse" to double the cursor speed when making a long impulse.
12. The "Stop at screen border" option enables the cursor to be snapped to the screen border you are getting nearer. Otherwise, the cursor will reappear on the opposite screen border.
13. Click **Close** when satisfied with your parameters.



The **Default** button lets you reset the TopHat options to the default parameters.

The Walk mode options slightly differ from the Fly mode options as shown below:



The Walk Sensitivity area lets you set the speed while moving along the X and Y axes and while rotating around the Z axis.

You can use the Vertical Translation area to choose whether you wish to use the buttons or the slider. In case you use the slider, you can indicate the maximum slider speed in the Max Speed field.

If you choose to use the buttons, clicking Go Up or Go Down then a joystick button lets you assign this button to the corresponding direction.

Note that all these information are saved in settings files.



Navigating in Examine Mode



This task shows you how to navigate in examine mode

Navigating in Examine Mode is the default mode. In this mode your object is driven by the joystick, it is just as if you would be using remote control to position your object. You are manipulating the object directly.

While manipulating, you can also examine your document as you would from the outside by moving around the document's perimeter, or as you would from within, turning your head to view or moving closer (zoom in, zoom out) to different objects.

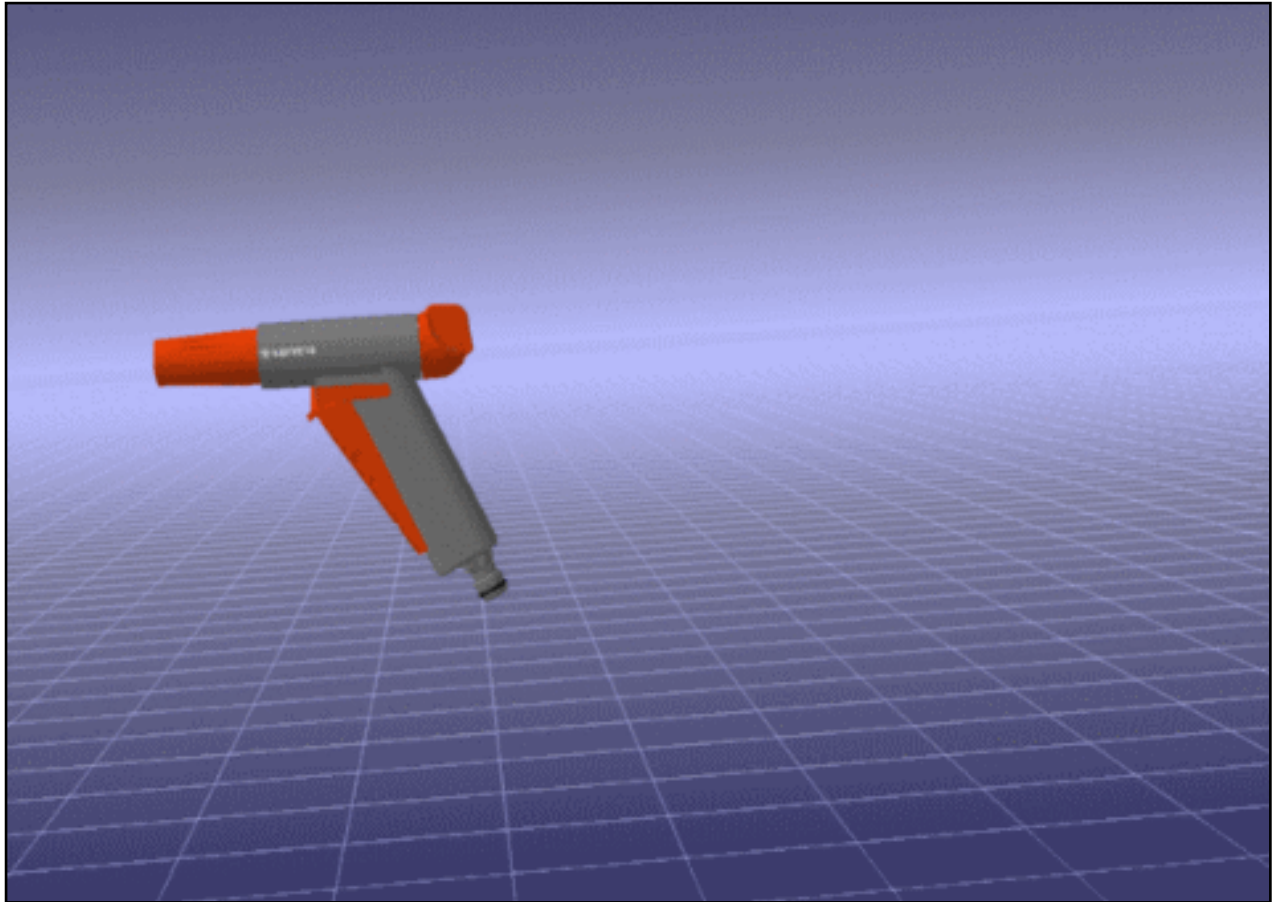


Open the [GARDENA.CATProduct](#) document.



1. Target

Point a location and click the side button and release (simple click). This selected location will be positioned and centered in the screen plane.

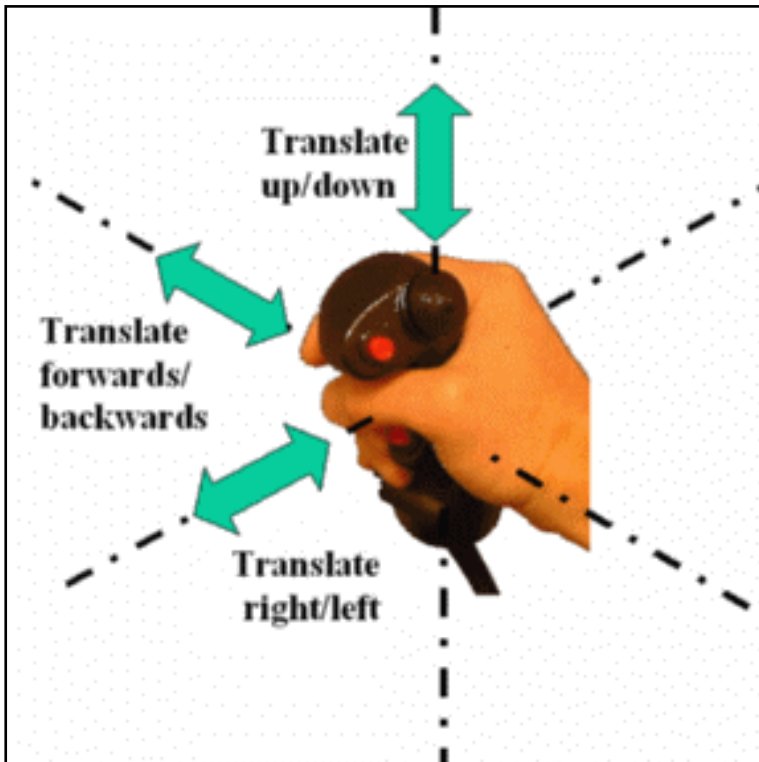


2. 3D Panning and Rotating

Click the side button without releasing: you are turning around the target.

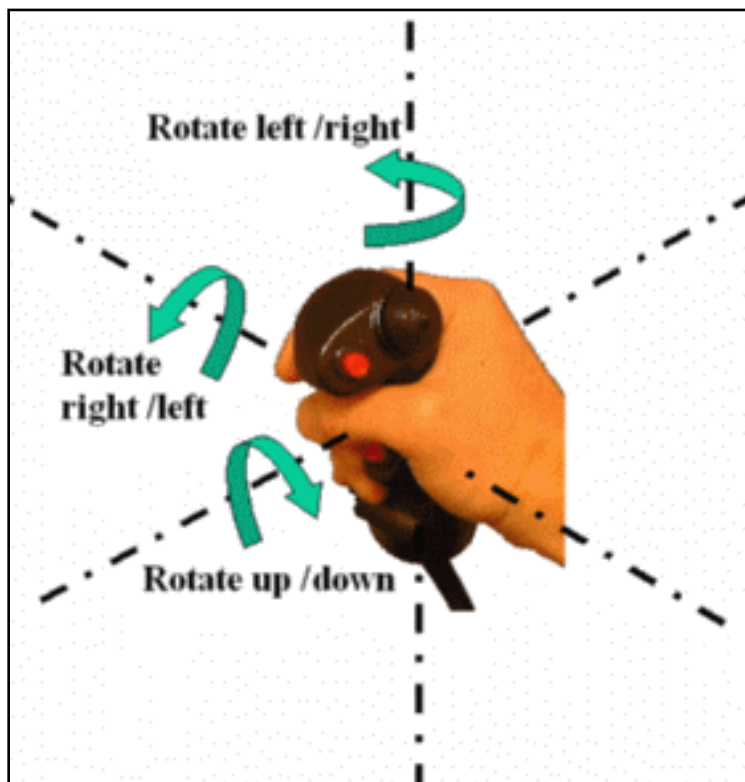
- **Translating**

Push the joystick forward or pull the joystick backward



- **Rotating**

Drag and /or rotate in 3D using the side button. See the figure below:



3. Rotating only

Click the side button without releasing and press the trigger without releasing (**still holding the side button down**).

4. Zooming in/ zooming out

Press and hold down the side button, then press and release the trigger button and push (to zoom in) or pull (to zoom out), still holding the side button down.

Zoom In



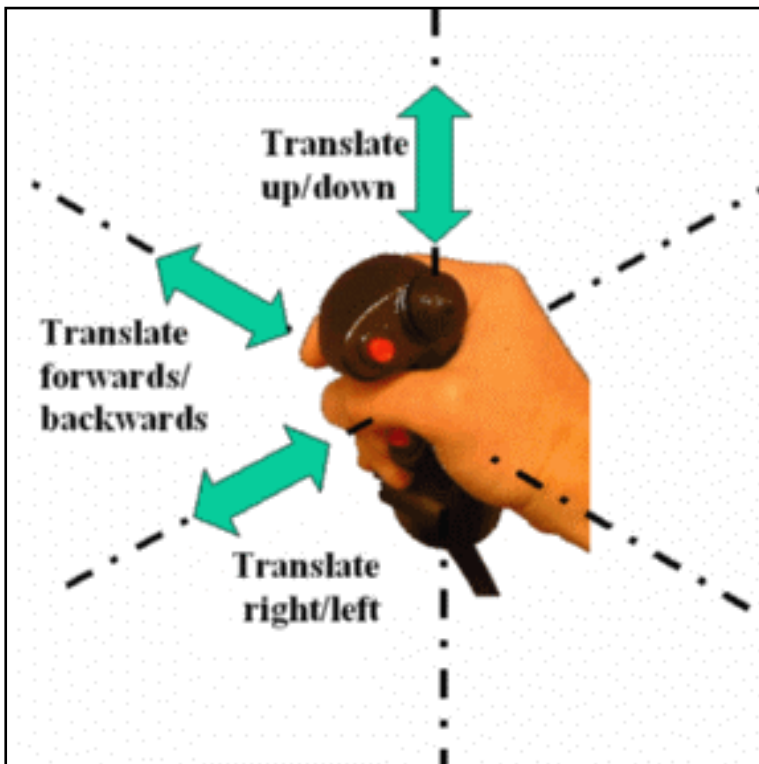
Zoom Out



5. Manipulating

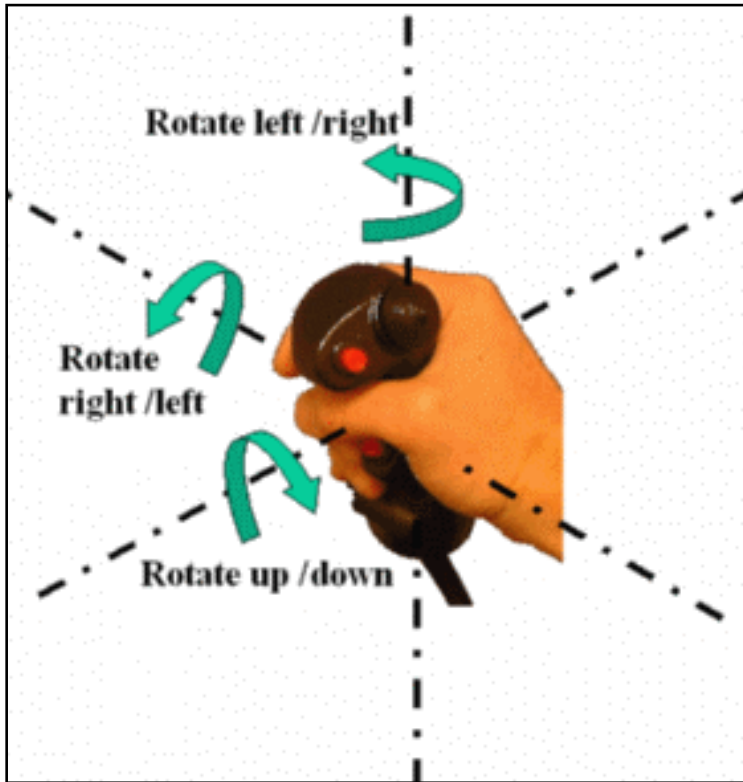
- **Translating**

Push the joystick forward or pull the joystick backward




- **Rotating**


Drag and /or rotate in 3D using the side button. See the figure below:




Navigating in Fly Mode




 In Fly mode you can move upward or downward on any horizontal view plane as you move forward or backward (backward in advanced mode only).

 This task shows you how to navigate in fly mode with a 3D joystick.


 You already opened your document [GARDENA.CATProduct](#) in DMU Navigator workbench (please refer to Loading documents in the ENOVIA - DMU Immersive review documentation).

 More About the joystick:

In Fly mode you will use the joystick **Side button**.

1. Click the Fly Mode  icon in the Immersive Review toolbar.

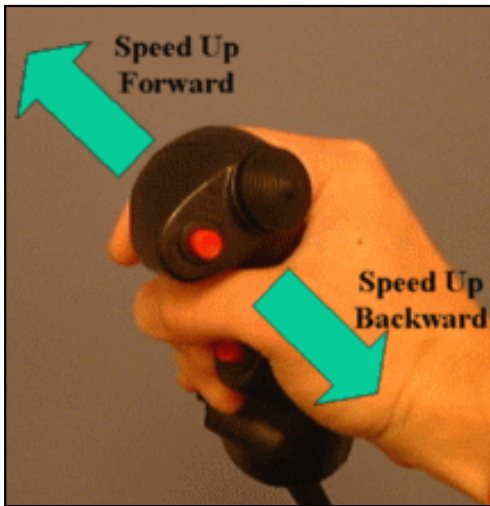
You can now begin to fly.

 Note that when the fly mode is activated, the head tracking behavior is still active. It means that you can still turn head during flying operation to look at details in a given direction.

2. Flying forwards/backwards

- if you push forward the joystick, the speed increases and you start moving forward (towards the circular symbol),
- if you pull the joystick, the speed decreases and you start moving backwards.

Note: Try not to incline the joystick performing this operation if you want to fly straightforward



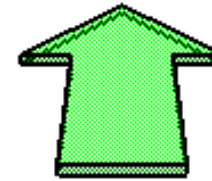
A green arrow appears along with a circular target located at the center of the view:

The speed at which you first approach the object is 0.

The further you push/pull the joystick from its starting position, the greater the speed.



This symbol represents the circular target towards which you fly.

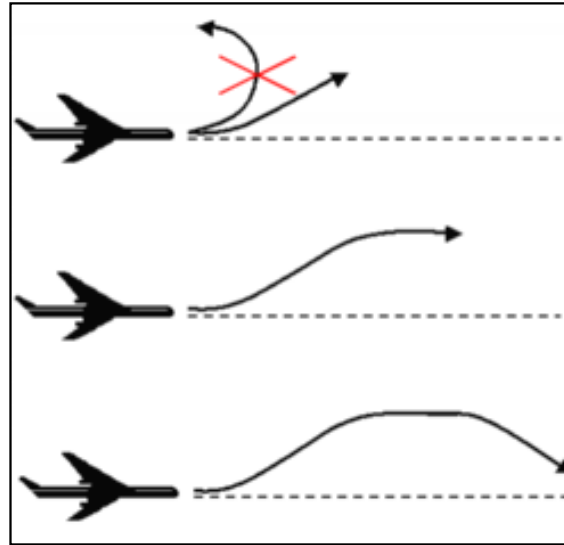


0.02

The figure below the arrow specifies the speed at which you are flying:

3. Flying up/down

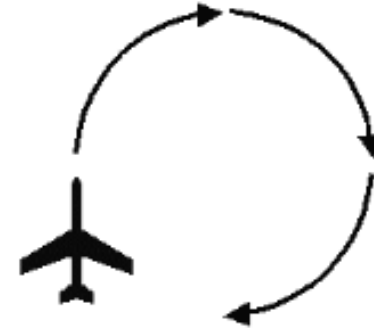
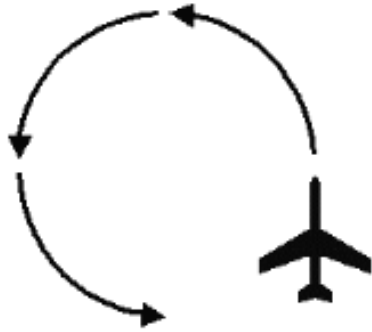
- if you incline the joystick forward, you start flying downward
- if you incline the joystick backwards, you start flying upward




If you pull the joystick at the maximum, you will not be able to perform a complete looping, you will just continue to fly up.
If you push the joystick back to its initial position, you will stop flying upward progressively.


4. Flying right/left

- If you turn left, you change direction (left)
- If you turn right, you change direction (right)



As you manipulate the joystick, the shape of the arrow changes to reflect the direction in which you are flying:

 You can combine simultaneously these joystick manipulations (above described) to obtain the fly behavior you wish

To return to the default navigation mode, click the Examine mode icon .



Installation Requirements



- [Basic Software Requirements](#)
- [Hardware Requirements](#)
- [How Are Version 5 Products Packaged?](#)
- [More About the Licensing Mechanism](#)

Basic Software Requirements



Common Software Requirements

Refer to the Program Directory or contact your IBM Support Center, for appropriate corrective service to apply to the software described in the topics that follow.

For more detailed installation requirements, refer to your *Installation and Administration Guide*.

Note: Windows 98 and Windows NT are no longer supported.

Windows 2000/XP

The following components at the minimum indicated level are required:

- Microsoft Windows XP Home Edition, Microsoft Windows XP Professional and Microsoft Windows 2000 Professional, with the following components:
 - Microsoft Windows 2000 delivers an implementation of OpenGL libraries. These libraries may be updated depending on the selected graphic adapter, when installing the graphic adapter and associated drivers. Dassault Systèmes will provide recommendations related to driver levels based on certified configurations through the following website at:

<http://www.ibm.com/solutions/engineering>
 - a localized version of the operating system may be required when the selected installation locale differs from Latin1 (for example, for the Japanese language environment)

Note: For remote access from networked clients, Terminal Server is available with Windows 2000 Server and Windows 2000 Advanced Server.

IBM AIX

The following components at the minimum indicated level are required:

- AIX Version 4 Release 3.2 or 3.3, including:
 - C Set++ for AIX Application Runtime:
 - at minimum level 3.6.4 for AIX 4.3.2 (5648-A81)

- at minimum level 4.0.2 for AIX 4.3.3 (5765-D52)

(C Set++ Application Runtime is shipped with AIX Operating System).

- IBM XL Fortran Runtime Environment for AIX (5765-C11, or 5801-AAR-7070, P/N 04L2123), at minimum level 5.1.0
- OpenGL and GL3.2 Runtime Environment (delivered with AIX 4.3 operating systems)
- CDE (Common Desktop Environment, delivered with the operating system).

HP-UX

The following components at the minimum indicated level are required:

- HP-UX Version 10.20 A.C.E. 4 (Workstation Additional Core Enhancements for HP-UX 10.20 - July 1999), or 10.20 A.C.E. 5 (Workstation Additional Core Enhancements for HP-UX 10.20 - December 1999), or HP-UX 11.0 A.C.E. (November 1999), including:
 - ANSI C++ Runtime Environment (aC++, at minimum level 1.21, delivered with the operating system)
 - HP FORTRAN Runtime Environment (delivered with the operating system)
 - HP-UX 700 OpenGL 3D API Runtime Environment
 - CDE (Common Desktop Environment, delivered with the operating system)
 - A localized version of the operating system may be required when the selected installation locale differs from ISO code pages.

Note: Because of binary incompatibilities between HP-UX 10.20 and HP-UX 11.0, support of HP-UX 11.0 is limited to a strict run-time environment.

SGI IRIX

The following components at the minimum indicated level are required:

- IRIX 6.5.2m, including:
 - C, C++ and Fortran77 standard execution environment (delivered with the operating system)
 - OpenGL (delivered with IRIX execution environment)
 - IRIX Interactive Desktop (delivered with the operating system)
 - WorldView is required when the selected installation locale differs from ISO-1.

Sun Solaris

The following components at the minimum indicated level are required:

- Sun Solaris 2.6.0, Solaris 7 or Solaris 8, including:
 - C and C++ runtime environment (delivered with the operating system)
 - OpenGL runtime environment (delivered with the operating system)
 - the Fortran runtime environment is delivered with Version 5
 - CDE (Common Desktop Environment, delivered with the operating system)
 - a localized version may be required when the selected installation locale differs from ISO-1.

Hardware Requirements



V5 Virtual Reality Infrastructure: supported hardware

- Workstations: SUN, IBM, HP, SGI, Windows
- multi-channel configurations necessary for some output devices (eg CAVE) are only supported on SGI multi-pipe workstations
- head mounted devices that are compatible with SGI multi-pipe workstations or Cyviz XPO Box are supported
- benches from Fakespace WorkDesk and Solutions ImmerseDesk Family, BARCO Baron
- dome from eLumen display devices
- active or passive stereo glasses: CrystalEyes and NuVision 60Gx
- 3D sensors for viewpoint tracking: Polhemus Fastrack Family and InterSense IS900
- 3D input devices: SpaceStick from Virtual Presence, Fakespace NeoWand, InterSense Wand

DMU Immersive Review: integrated hardware

- Workstations: SUN, IBM, HP, SGI, Windows
- multi-channel configurations necessary for some output devices are only supported on SGI multi-pipe workstations
- all benches configurations supported in V5
- active or passive stereo glasses supported in V5
- 3D sensors for viewpoint tracking supported in V5
- 3D input devices: SpaceStick from Virtual Presence

How Are Version 5 Products Packaged?



The Version 5 product packaging model is based on the concepts of configurations and products.

Configurations

Configurations are a convenient and attractive way for you to order and install the adequate combination of products for each type of user, while offering a single solution from a licensing point of view.

There are two types of configurations:

- **standard configurations** contain a pre-defined list of products, corresponding to most frequent user profiles across industries and processes. These configurations are offered at an attractive price compared to the sum of the individual product prices
- the content of **custom configurations** is dynamically defined at ordering time, thus allowing you to adapt the configuration content to the most specific user needs. The content of a custom configuration is defined by adding individual products (see product delivered as "add-on" below) to an existing standard configuration. The result is a competitively priced solution, and remains a single solution from a licensing point of view.

After initial installation, the configuration mechanism lets you manage the evolution and growth of your user profile content by allowing you to add new products. The resulting new seat definition is still a single solution from a licensing point of view.

To be able to use Version 5, you need to purchase and acquire at least one configuration license.

NOTE: you need the DN3 configuration to access the DMU immersive Review Product (Du3).

If you already have a custom configuration, you can extend it by adding products. But before you do so, you must use LUM to migrate your server license database to support custom configuration growth.

To do so:

- stop your LUM license server
- migrate your server license database to the new format using the command:

i4ccmig

- then import your new license as usual.

Products

Products are the elementary software building blocks for Version 5 installations.

Version 5 may be ordered in three ways:

- As a standard configuration
- As an "add-on" product on top of a standard configuration to build a custom configuration
- As a "shareable" product. In this case the product is delivered with its own license key, allowing the user to obtain the license at the beginning of the session, or to leave it for another user. Prices of products ordered in this mode are different, versus "add-on" price, to take into account multiple users potential. Shareable product licenses do not have serial numbers.

More About the Licensing Mechanism



Version 5 delivers identical licensing mechanisms on UNIX and Windows environments, based on LUM (License Use Management). The following licensing principles apply:

- Using a given Version 5 product requires a license for it and for its prerequisite products
- Using a given Version 5 configuration requires a license for it. Licenses for Version 5 configurations are acquired and released for the total configuration. The products within a configuration cannot be shared
- In all cases, licenses are acquired at the beginning of the process, and released at its termination.

Version 5 can be used in two licensing modes: nodelock or with concurrent usage of licenses on a network.

Nodelock Licensing

The use of local display of the hardware configuration is mandatory for Version 5 usage in nodelock mode.

There is no limit to the number of Version 5 processes launched for a given license (product or configuration). For instance, a user can launch the following simultaneous processes:

- a Version 5 interactive session
- a Version 5 process executed through an OLE container application
- replay of macros recorded from captured sequences of Version 5 user interactions.

Concurrent Licensing

A user on one machine, using one display, uses one license per product used, regardless of the number of processes. If the display changes, then an additional license is taken for the corresponding process.

Add-on and shareable products require a license for a configuration which includes at least the prerequisite products. Licenses for Version 5 configurations are acquired and released for the total configuration. The functions within a configuration cannot be shared.

Demo Usage

In addition to its normal mode of operation where all licensed functions are accessed, Version 5 is capable

of running in demo mode, on UNIX and Windows, with some disabled functions (such as **File->Save** - see list below):

- Existing Version 5 customers, who have a minimum of one regular license, can switch from standard mode to demo mode (**Tools->Options->Licensing** tab). As the user restarts a session, the demo mode will be automatically used
- Qualified prospects, who may be given the Version 5 code for evaluation purposes, are required to enter a special demo license key. This will ensure that the code starts automatically in demo mode.

With this mechanism, customers can explore add-on products for which they do not yet have a license. The qualified prospect can get first hands-on experience, verify the ease of use of Version 5, and create the first parts. In both cases, a favorable business environment is created for accelerating sales cycles.

When using Version 5 in demo mode, the following functions are disabled:

- File Save and Save as
- File Read (except for prepared Version 5 demo documents)
- Embedding Version 5 documents in OLE documents
- Opening Version 5 documents using OLE technology
- Cutting, copying and pasting Version 5 documents with the Windows clipboard
- Recording and replaying macros.

Using and Customizing Fonts

About Fonts

Customizing User Interface Fonts on Windows

Customizing User Interface Fonts on UNIX

Customizing Fonts for Displaying Texts


Customizing Fonts for Displaying Geometry Area Texts

Adding Extra PostScript Fonts

Understanding Differences Between Font Display and Printed Output

Recovering Custom Fonts Developed Using CATIA Version 4

About Fonts

 This section contains principally conceptual information about font support in general, identifies which areas of the Version 5 software are concerned by font support, and explains how you can customize fonts.

The areas of the software which allow font customization are:

- user interface: menu names, command names, tooltips, dialog box names and texts, etc.
- specification tree texts
- texts you enter in certain applications, and which reference fonts: a typical example is the text you enter in drawing documents created using the Generative Drafting product.

You will also find information about how, in certain contexts, the text you see in the geometry area may not look exactly the same when you print.

And finally, if you used the CATFONT utility in CATIA Version 4 to customize your own fonts, you will also find information about how to recover the fonts for use in Version 5.



Customizing User Interface Fonts on Windows

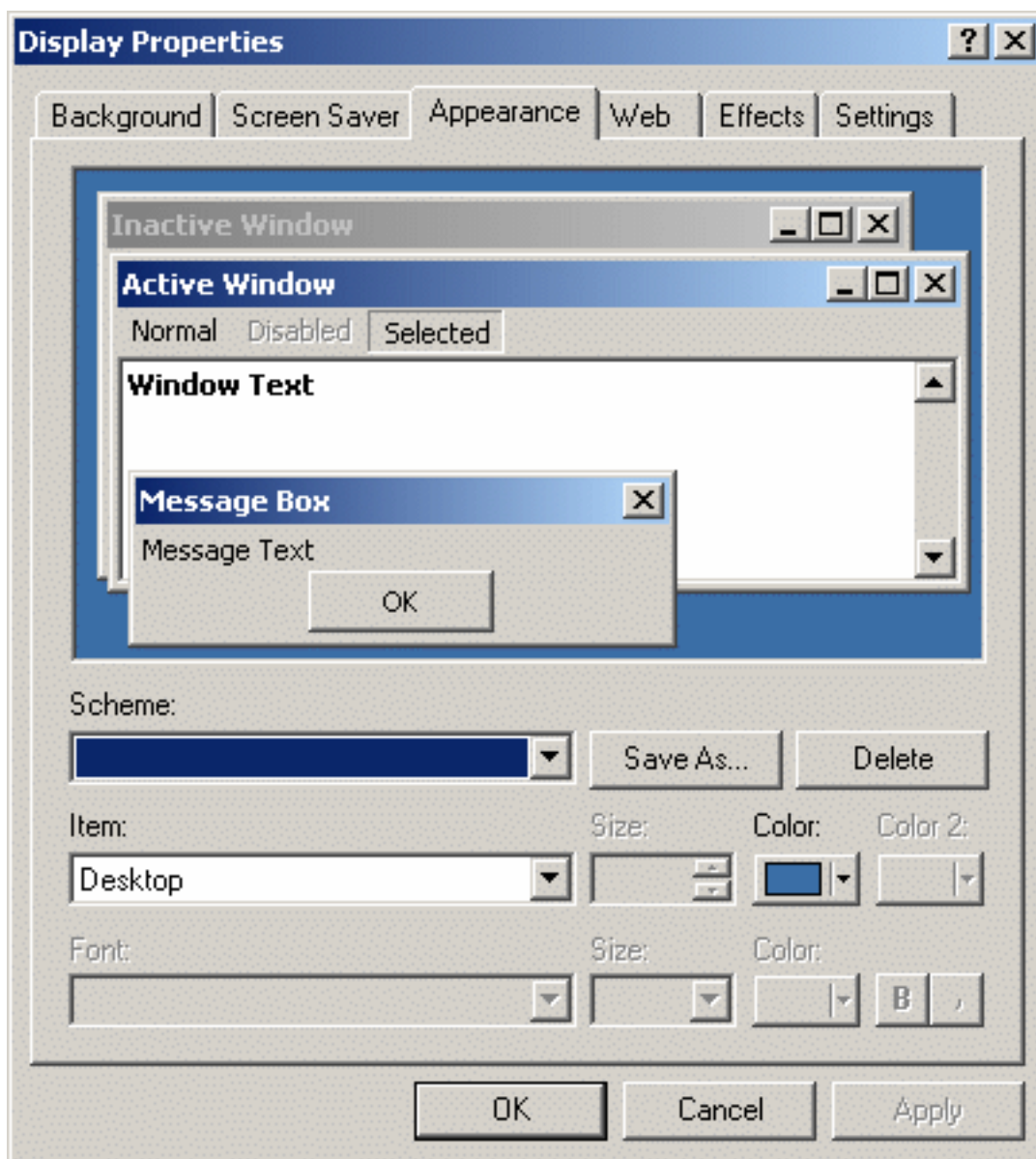


This task explains how to customize user interface fonts on Windows.



1. Select the **Start->Settings->Control Panel** command, then double-click Display and select the Appearance tab (if it is not displayed by default).

The following dialog box appears:



2. Use the Item list to select an item of the user interface you want to customize, or click on a user interface item in the area in the center of the dialog box.



For example, select the items:

- Menu
- Message Box
- if you want to customize menu command, message box, dialog box and tooltip text fonts.

3. Select the desired font, font size and color.

4. Click **Apply**, then **OK**.

5. If you have a Version 5 session open, exit the session and restart to see the changes take effect.



Customizing User Interface Fonts on UNIX



This task explains how to customize user interface fonts on UNIX.



1. Go to the directory:

`Install_directory/resources/msgcatalog/`

where "Install_directory" is the root installation directory you chose when installing the software.

If you installed the software at the default location, you would go to the following directory:

`/usr/Dassault Systemes/B13/OS_a/resources/msgcatalog`

if you are using the English language. A subdirectory is provided for each language supported. Go to the appropriate subdirectory if you are using a language other than English.

2. Edit the file named "Dialog".

The "Dialog" file contains resource declarations for fonts (and foreground and background colors) for certain user interface components. The file is delivered at installation and is ready for use as is.

Note that you can declare Motif fonts only.

3. Customize the last line of each user interface component declaration if you want to change the font and the font size.
4. If you have a Version 5 session open, exit the session and restart to see the changes take effect.

Make sure you exported the LANG variable for the desired locale before restarting a session.



Customizing Fonts for Displaying Texts



This task explains how to customize the fonts used, for example, to display:

- texts in the specification tree
- constraint texts in the Sketcher, Part Design, Assembly workbenches

and, in general, all 2D texts. Note that system fonts are used for displaying these types of text.

This does not concern texts you enter in drawing documents created using the Generative Drafting product.

On Windows



1. Select the **Start->Settings->Control Panel** command, click the Display control then click the Appearance tab (if it is not displayed by default).
2. In the Item list, select the Message Box item, or click inside the Message Box item (on Message Text) in the area in the center of the dialog box.
3. Select the desired font, font size and color.
4. Click **Apply**, then **OK**.
5. If you have a Version 5 session open, exit the session and restart to see the changes take effect.

On UNIX



1. Go to the directory:

`Install_directory/resources/msgcatalog`

where "Install_directory" is the root installation directory you chose when installing the software.

If you installed the software at the default location, you would go to the following directory:

`/usr/Dassault Systemes/B13/OS_a/resources/msgcatalog`

if you are using the English language. A subdirectory is provided for each language supported. Go to the appropriate subdirectory if you are using a language other than English.

2. Edit the file named "Visualization".

The "Visualization" file contains resource declarations for fonts for displaying annotation texts. Each font is declared using six identical declarations which differ only in the font size. This is required to allow end users to zoom the text size. The range of font sizes allows end users to zoom within the limits of those font sizes.

The file is delivered at installation and is ready for use as is.

Note that you can declare Motif fonts only.



The Visualization file for the DBCS language (Japanese, Korean and Simplified Chinese) environments contains two lines for each declaration: one line for the SBCS character set, and one line for the DBCS character set.

Customize each line if you want to change the font.

3. If you have a Version 5 session open, exit the session and restart to see the changes take effect.

Make sure you exported the LANG variable for the desired locale before restarting a session.



If you exchange documents with a site using a different language, text display may contain "garbage" in 2D areas such as the specification tree and in editable fields. As explained above, specification tree texts, for example, are displayed using system fonts. If you do not have the same fonts installed on both sites, the text will not be lost: it will just not be displayed correctly.

To display all text correctly, you must do the following on the receiving site:

- install and activate the appropriate locale for reading the document
- make sure that the appropriate system fonts are installed and declared correctly on the receiving site
- Text in the geometry area points to CATIA Version 4, Bitstream or custom fonts. Make sure that the appropriate fonts are installed and declared correctly on the receiving site.



Customizing Fonts for Displaying Geometry Area Texts



This task explains how to choose fonted texts displayed in the geometry area, for example, when using the Generative Drafting product, and lists the fonts you can choose from.



You need access to the Generative Drafting product license to follow this scenario which shows you how to enter text in a drawing and choose a font for the text. The objective is to present the list of fonts supported.



1. Open a drawing you created using the Generative Drafting product.



2. Click the  icon, then click a point in the drawing to position the text.

The Text Editor dialog box is displayed.

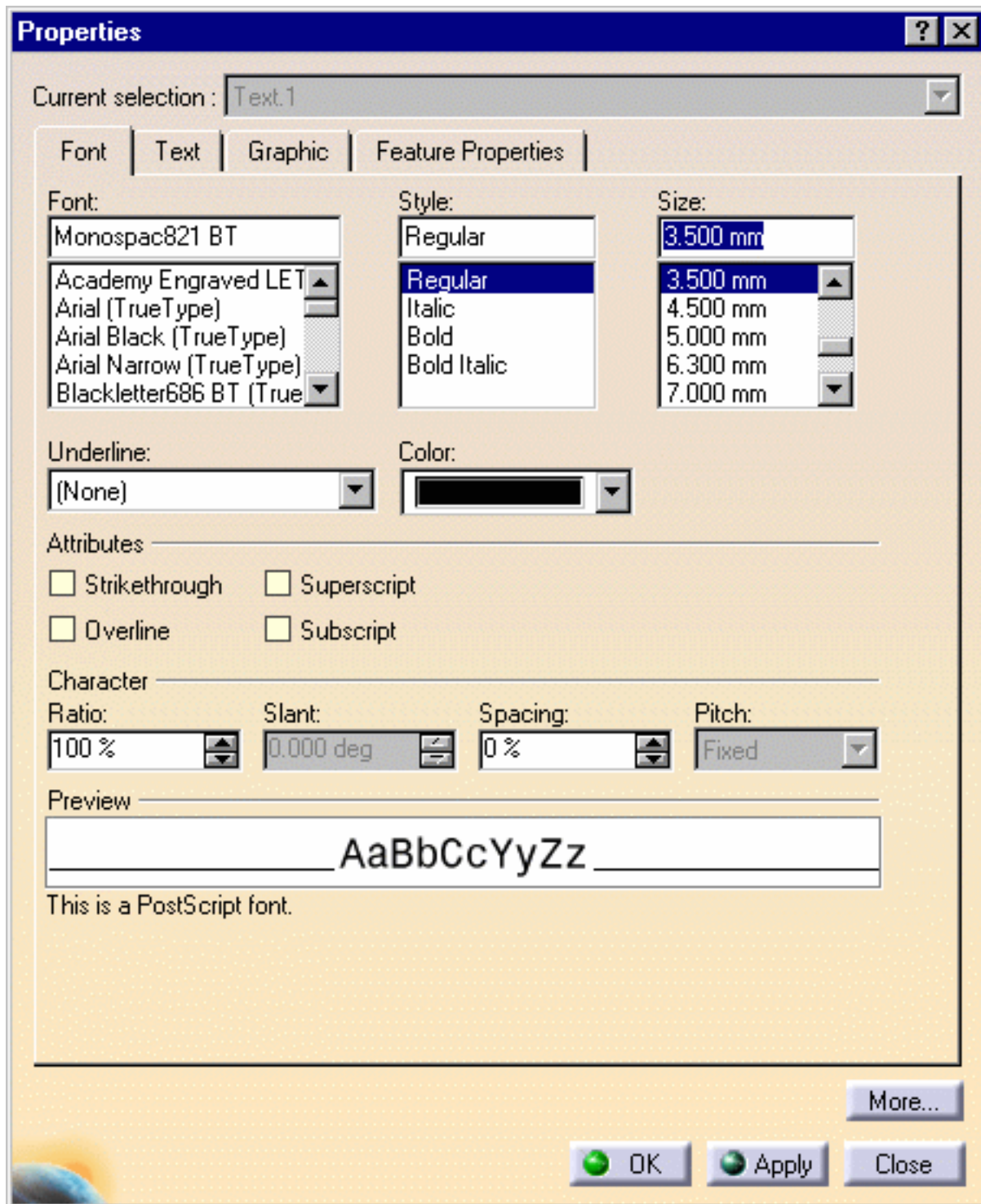
3. Use the Text Editor dialog box to write the text, justify it, specify the text height and define the anchor point, then click anywhere outside the Text Editor window, but inside the main application window.

The Text Editor dialog box disappears.

4. Point to the text, right-click and select the Properties command.

The Properties dialog box is displayed:

5. Click the Font tab. The Font tab includes controls for setting the font, font style and size:



Which Font Formats Are Supported?

Version 5 provides the following font formats:

- PostScript Type 1 format
- CATIA Version 4 FONT format
- TrueType format

With respect to the PostScript Type 1 font format, note that on Windows only, if a TrueType version of the font exists or has been created, the TrueType version of the font can be used to optimize visualization quality.



Which Fonts Are Provided?

The following fonts are supported and are installed ready for use without further customization when you install Version 5:

- all default stroke fonts delivered previously with CATIA Version 4
- 22 Bitstream Type 1 fonts
- an extra font (customized by Bitstream): CATIA Symbols; this font contains the symbols from Version 4 fonts
- TrueType fonts provided by Windows.

Note that the Bitstream fonts are delivered in several different styles (depending on the font), whereas the CATIA Version 4 fonts are delivered in regular style only.

Furthermore, the 22 Bitstream fonts support ISO-8859-1 environments only.

The fonts in TrueType format may be used as is, i.e. all fonts delivered will be displayed in the font list. However, you can customize this list (for instance, to keep only the fonts you use most frequently) by copying the desired fonts to your Version 5 environment in:

`install_root\resources\fonts\TrueType`

where "*install_root*" is the installation folder (Windows).

For TrueType fonts, a ".ttf" file is required.

Which Bitstream Fonts Are Supported?

The Bitstream fonts are:

Font Name	Attribute	File Name
Swis721 BT	roman	Swiss.pfb
	italic	SwissI.pfb
	bold	SwissB.pfb
	bold italic	SwissBI.pfb
Swis721 LtCn BT	light condensed	SwissCL.pfb
	light condensed italic	SwissCLI.pfb
Swis721 BdOulBT	bold outline	SwiOuB.pfb
Monospac821 BT	roman	Monos.pfb
	italic	MonosI.pfb
	bold	MonosB.pfb
	bold italic	MonosBI.pfb
Dutch801 Rm BT	roman	Dutch.pfb
	italic	DutchI.pfb
	bold	DutchB.pfb
	bold italic	DutchBI.pfb
Courier10 BT	roman	Coure.pfb
	italic	CoureI.pfb
	bold	CoureB.pfb
	bold italic	CoureBI.pfb
UniversalMath1 BT	regular	Mathe.pfb
SymbolMono BT	regular	SymbM.pfb
SymbolProp BT	regular	SymbP.pfb

Note that:

- the Swiss 721 Bitstream font family is Bitstream's version of Helvetica
- the Monospace 821 Bitstream font family is Bitstream's version of Helvetica Monospaced
- the Dutch 801 Bitstream font family is Bitstream's version of Times Roman
- the CATIA Symbols font (not in the above list) contains the symbols from Version 4 fonts.

For each of the Bitstream fonts, the following files are delivered in the location referenced by the CATFontPath variable:

- in the Postscript folder or subdirectory: .pfb, .inf, .pfm, .afm
- in the ExtraFiles folder or subdirectory: .ttf. Note: On Windows only, installing Version 5 also installs in the ExtraFiles environment the equivalent fonts in TrueType format. The TrueType font format offers enhanced visualization quality. The installation adds the fonts (in TrueType format) to the list of system fonts you can view by selecting the **Start->Settings->Control Panel** command and double-clicking the Fonts control.

Which Version 4 Fonts Are Provided?

The following Version 4 fonts are supported and are installed ready for use without further customization when you install Version 5:

- SSS1.font, SSS2.font, SSS3.font, SSS4.font: 4 simplex sans serif fonts
- ROM1.font, ROM2.font, ROM3.font: 3 roman fonts
- GOTH.font: 1 Gothic font
- SYM1.font, SYM2.font, SYM3.font, SYM4.font: 4 symbol fonts
- KANJ.font: Kanji font (Japanese)

Regarding the KANJ font, from now on, halfwidth Katakana characters are displayed with a smaller width than the width with which they were displayed in CATIA Version 4


- KOHG.font: Hangeul font (Korean)
- TRCH.font: Traditional Chinese font
- SICH.font: Simplified Chinese font.

Note that:

- SYM1 contains annotation and tolerance symbols, and plot markup characters
- SYM2 contains ISO symbols fonts
- SYM3 contains roughness symbols
- SYM4 contains graphic and mathematical symbols as well as miscellaneous technical symbols.



Adding Extra PostScript Fonts

 This section explains how to add extra PostScript Type 1 fonts. A large choice of PostScript or TrueType fonts is available from Bitstream Inc. (<http://bitstream.com>). You can address questions about ordering Bitstream fonts to:

http://catia_support@bitstream.com



1. Copy the new fonts to your Version 5 environment in:

install_root\resources\fonts\PostScript

where "*install_root*" is the installation folder (Windows) or directory (UNIX).

For Type 1 fonts, the following files are required:

- ".pfb" or ".pfa" font files
- ".afm" and ".inf" files.

To achieve enhanced font visualization (Windows only), if the associated TrueType font does not exist, you can ask the system to generate it from the PostScript files (the following file types are needed: .pfb, .afm, .inf).

2. To do so, select the **Start->Settings->Control Panel** command, double-click the Fonts control, then select the **File->Install->New Font...** command.
3. Select the font, and when prompted, check the three options, notably the first option in the dialog box: "Convert Type1 to TrueType", then click **OK**.

If an agreement exists between the font supplier and Microsoft, the system will then generate the corresponding TrueType font. If not, the font will not be generated.

4. Edit the file:

install_root\resources\fonts\PostScriptRelatedTrueType


to map the PostScript file name with the full name of the TrueType equivalent. The "full name" refers to the name of the font visible when selecting the **Start->Settings->Control Panel** command and double-clicking the Fonts control.



If you already have the TrueType fonts, you can simply install them. To do so, select the **Start->Settings->Control Panel** command, double-click the Fonts control, then select the **File->Install->New Font...** command and select the fonts to be installed.



Understanding Differences Between Font Display and Printed Output

 This section contains information about minor differences you may notice between the way text is displayed in the geometry area and how it looks in printed output.

Text Display

In general, there is a slight difference in how text is displayed between Windows and UNIX: on Windows, text is displayed using TrueType fonts, whereas Type 1 Postscript fonts are used on UNIX.

Differences between Text Display and Printed Output


When printing on printers using the drivers listed below, you can expect the following results:


Driver	Font Used for Printing On Windows	Font Used for Printing On UNIX
GDI	<p>TrueType</p> <p>Any differences between displayed and printed text depend on how your System Administrator set up your printer.</p> <p>Your printer may be set up in one of three ways:</p> <ul style="list-style-type: none"> the printer has all the required memory and fonts 	<p>N.A.</p> <p>Discretization is applied to Drafting texts.</p> <p>For 2D texts (for example, in the specification tree), if the CATIA - P2 setting is active, printing the text produces a bitmap, whereas in all other cases, texts are printed using the font referenced in the following resource file:</p>
PostScript	<ul style="list-style-type: none"> the printer does not have the fonts, but the PostScript file contains fonts or you may have to map your printer to a specific font. For example, to view your printer properties in a Version 5 session, select the File->Print command, click the Properties... button and view the printer properties on the Advanced tab. 	<p><code>Install_folder/resources/msgcatalog/Print</code></p> <p>By default, the font is Helvetica.</p> <p>For DBCS languages, discretization is applied to 2D texts.</p>

CGM HP-GL2,
HP-RTL CalComp
C907 Océ
Graphics GPR50 N.A.
Versatec: VCGL
and VGS 2.0
VRF

Discretization is applied to texts

Recovering Custom Fonts Developed Using CATIA Version 4

 This task explains how to use CATIA Version 4 FONT files and set them up in your Version 5 environment. For more information about CATIA Version 4 FONT files, refer to your CATFONT Version 4 utility documentation.

-  **1.** Make sure that the fonts you created are in the FONT format (described using the UNICODE code key) and not in the FONTDATA format (described using the EBCDIC code key).

The internal font format we recommend from CATIA Version 4 Release 1.8 onwards is described using UNICODE codes.

If you did not already migrate your user FONTDATA files to FONT files and FONT CODE files, you must do so using the CATFONT utility using, as a minimum level, CATIA Version 4 Release 1.8. Regarding FONT CODE names, refer to your CATFONT utility documentation for more details.

Note that the following is no longer supported:

- the font described with proportional format
- the grid defined by five numbers.

In both cases, the associated orders are ignored. Note that none of the CATIA Version 4 basic delivered fonts used any of these options.

- 2.** Copy the Version 4 FONT files to your Version 5 environment in:

install_folder\resources\fonts\Stroke

where "*install_folder*" is the installation folder (Windows) or directory (UNIX).

- 3.** Copy the FONT CODE files to your Version 5 environment in:

install_folder\reffiles\NLS\fontcode

4. Rename these FONT CODE files using the following syntax:

`XXXX.fontcode` renamed to `FCUSER n .fontcode`

where "`XXXX.fontcode`" represents the V4 font code file and n the increment (16 being the maximum number).

For instance:

`ABBK.fontcode` renamed to `FCUSER1.fontcode`
`HEL1.fontcode` renamed to `FCUSER2.fontcode`
`HEL2.fontcode` renamed to `FCUSER3.fontcode`
`SPEC.fontcode` renamed to `FCUSER4.fontcode`
`TIME.fontcode` renamed to `FCUSER5.fontcode`

Note: the names `FCUSER1` to `FCUSER12` are reserved for SBCS font codes whereas `FCUSER13` to `FCUSER16` are reserved for DBCS font codes.

5. Edit the *V4FontInteroperability* file in:

`install_folder\resources\fonts\`

by adding your Version 4 FONTLIB names to the list.

This file maps to a Version 4 FONTLIB name, the FONT and FONT CODE associated with it.

The example below shows two fonts (ABBK and TIME) added to *V4FontInteroperability* with their corresponding font code:

FONTLIB Font		Encoding	Flag for Mainframe data Conversions
IS01	SSS2.Font	CPXI1.codepage	new
IS0W	SSS4.Font	CPXI1.codepage	new
IS02	SSS2.Font	CPXI2.codepage	new
IS05	SSS2.Font	CPXI5.codepage	new
I850	SSS2.Font	CPIA850.codepage	new
S932	SSS1.Font	FCS932.Fontcode	new
KANJ	KANJ.Font	CPIA932.codepage	old
KANK	KANJ.Font	CPIA932.codepage	old
KOHG	KOHG.Font	CPEKR.codepage	old
SICH	SICH.Font	CPECN.codepage	old
TRCH	TRCH.Font	CPIA938.codepage	old
DSUS	SSS1.Font	CPDAUS.codepage	old
DSDT	SSS1.Font	CPDADT.codepage	old
DSFC	SSS1.Font	CPDAFR.codepage	old
DSUK	SSS1.Font	CPDAUK.codepage	old
DSJP	SSS1.Font	FCDSJP.Fontcode	old
DSIT	SSS1.Font	FCDSIT.Fontcode	old
DSSD	SSS1.Font	CPDASM.codepage	old
DSHB	SSS1.Font	FCDSHB.Fontcode	old
DSDC	SSS1.Font	FCSDSC.Fontcode	old
STDD	SSS3.Font	CPDAUS.codepage	old
ROM1	ROM1.Font	CPDAUS.codepage	old
ROM2	ROM2.Font	CPDAUS.codepage	old
ROM3	ROM3.Font	CPDAUS.codepage	old
GOTH	GOTH.Font	CPDAUS.codepage	old
LINE	LINE.Font	CPDAUS.codepage	old
DSFT	SYM1.Font	FCDSFT.Fontcode	old
DSIS	SYM2.Font	FCDSIS.Fontcode	old
DSES	SYM3.Font	FCDSIS.Fontcode	old
SET9	SYM4.Font	FCSET9.Fontcode	old
SF01	SYM4.Font	FCSF01.Fontcode	old
SF02	SYM4.Font	FCSF02.Fontcode	old
SF03	SYM4.Font	FCSF03.Fontcode	old
ABBK	ABBK.Font	FCUSER1.Fontcode	old
TIME	TIME.Font	FCUSER5.Fontcode	old



Regarding Korean fonts, since there are differences in Korean ideogram UNICODE codes between the UNICODE used in Version 4 and Version 5, no Version 4 user-defined Korean font can be used directly in Version 5.



Conferencing

Initializing a Conference Infrastructure

Initializing a Conference on UNIX using the Backbone Driver As Backb

one Manager, launch the backbone daemon on node1: "**CATSysDemon -dm domain.lst -timeout 3000**". Launch the Backbone daemon on both node2 and node3 as follows: "**export CATBBDomainManager=node1 CATSysDemon -timeout 3000**". Once initialized, all users must select the **Backbone** driver option using: **Tools-> Options -> General -> General**. In the **Conferencing** area, check the **Backbone** option.

Initializing a Conference on Windows using the Backbone Driver As Backbone Manager, launch the backbone daemon on node1: "**CATSysDemon -dm domain.lst -timeout 3000**". Launch the Backbone daemon on both node2 and node3 as follows: "**export CATBBDomainManager=node1 CATSysDemon -timeout 3000**". Once initialized, all users must select the **Backbone** driver option using: **Tools-> Options -> General -> General**. In the **Conferencing** area, check the **Backbone** option.

Initializing a Conference on Windows using the NetMeeting Driver All users must select the **NetMeeting** driver option using: **Tools-> Options-> General -> General**. In the **Conferencing** area, check the **NetMeeting** option.

Organizing a Conference

Launching a Conference as Host In the menu bar, select the **Tools-> Conferencing-> Host**

Joining a Conference as Guest In the menu bar, select the **Tools-> Conferencing-> Guest**

Working in a Conference

Leading the Visual Conference

Sharing Documents

Transferring Files

Sending Messages to other Conference Participants

Customizing Conference Options

Consulting Conference History

Leaving a Conference

Using Knowledgeware Capabilities

Parameters

Formulas

Design Tables

The Knowledgeware Language

Working Through the Knowledgeware Capabilities

Integration with Enovia V5

Parameters

[Introducing Parameters](#)
[Creating a Parameter](#)
[Copy/Pasting Parameters](#)
[Specifying a Parameter Value as a Measure](#)
[Importing Parameters](#)
[Specifying the Material Parameter](#)
[Creating Points, Lines... as Parameters](#)
[Applying Ranges to Parameters by Using a Rule](#)
[Using Relations based on Publications at the Product Level](#)
[Creating an Associative Link between Measures and Parameters](#)
[Publishing Parameters](#)
[Activating and Deactivating a Component](#)
[Parameters: Useful Tips](#)



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

Introducing Parameters

- [Displaying Parameters in the Specification Tree](#)
- [Parameters and National Support Languages](#)
- [Editing a Parameter](#)
- [Hiding a Parameter](#)

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

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

User parameters are very handy in knowledgware applications:

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

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

Displaying Parameters in the Specification Tree

The user parameters are displayed in the specification tree provided you check the **Parameters** box in the **Display** tab in the **Tools->Options->Infrastructure->Part Infrastructure** dialog box. The user parameter list contains at least the Material parameter. The initial value of the Material parameter is set to None.

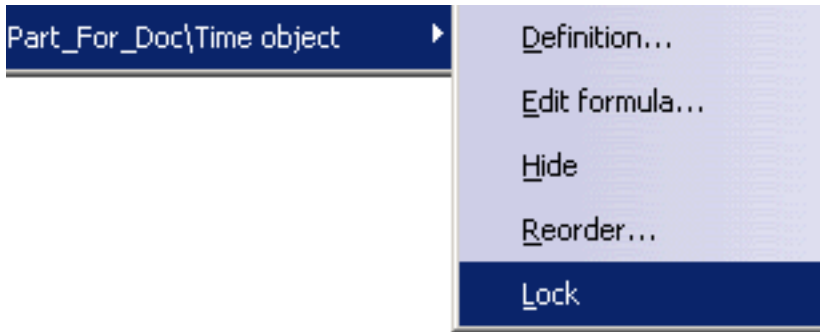
In addition, parameters can be displayed with their values provided you check the With Value box below the Parameter Tree View settings in the **Tools->Options->General->Parameters and Measure** dialog box

Parameters and National Support Languages

CATIA users working with non-latin characters should check the **Tools->Options->General->Parameters and Measure->Parameter Names->Surrounded by'** option. Otherwise, parameter names should have to be renamed in latin characters when used in formulas.

Editing a Parameter

You can access a parameter contextual menu by right-clicking this parameter in the specification tree.



The **Definition...** command enables you to access the Edit Parameter window where you can edit the parameter value.

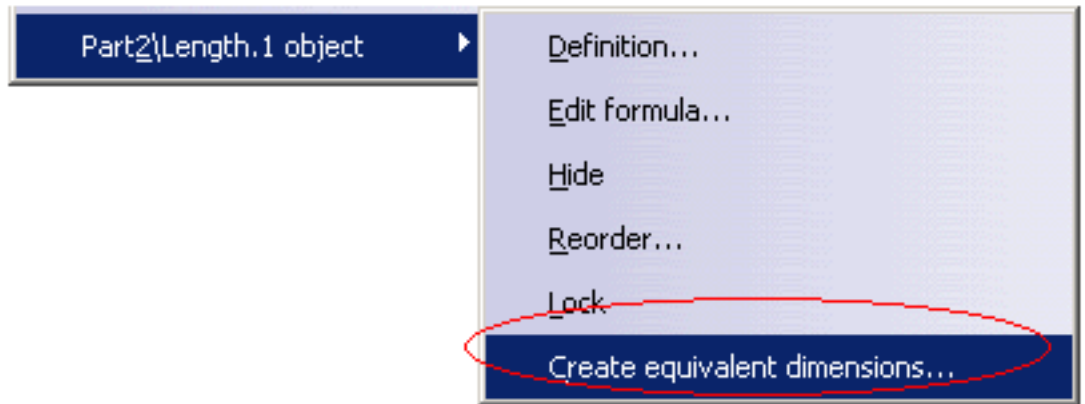
The **Edit formula...** command enables you to access the formula editor to add a formula to the parameter.

The **Hide** command enables you to hide the parameter. In this case, it will not display in the specification tree.

The **Reorder...** command enables you to reorder parameters.

The **Lock...** command enables you to lock the parameter.

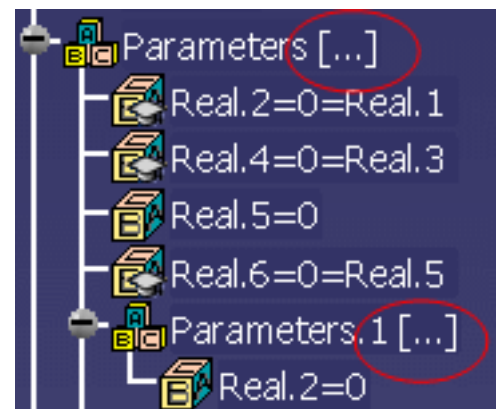
When working with parameters of Length type, you can also use the **Create Equivalent Dimensions** command. To know more about this command, see [Getting Familiar with the Equivalent Dimensions Interface](#) or [Using the Equivalent Dimensions Feature](#).



Hiding a Parameter

You can hide a parameter by right-clicking this parameter in the specification tree and by selecting the **Hide** command.

- A visual indicator located at the parameters set level indicates that the set contains hidden parameters. Note that this indicator is not recursive.
- If the user tries to delete a parameters set containing hidden parameters, a message displays asking the user if he wants to delete the parameters set that contains hidden parameters. Note that if the user tries to delete a parameters set containing another parameters set with hidden parameters, the message will display.




Creating a Parameter



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



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

Parameter	Value
PartBody\Hole.1\ThreadingDepth	20mm
PartBody\Hole.1\Activity	true
Mechanical_Property\Volume	0m3
Mechanical_Property\Mass	0kg
Mechanical_Property\WetArea	0m2
Material	None
Machining_Time	1000s

Edit name, value or formula

Machining_Time 1000s =

New Parameter of type Time With Single Value

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



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



Copy/Pasting Parameters

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

- Paste a parameter without the formula which defines it.
For example:
 $\text{Holeplus} = 15 = \text{Diameter} + 10$ will be pasted as $\text{Real.i} = 15$
(if the With Value box is checked)
- Paste a parameter as well as the formula which defines it, but only if the parameters referred to in the formula are also selected in the copy.
For example:
 $\text{Holeplus} = 15 = \text{Diameter} + 10$ will be pasted as $\text{Real.i} = 15$ if the Diameter parameter does not belong to the items selected for the copy
but HolePlus will be pasted as $\text{Real.i} = 15 = \text{Real.j} + 10$ if Diameter is selected in the copy (use multi-selection).
- Paste a parameter as well as the formula.
 $\text{Holeplus} = 15 = \text{Diameter} + 10$ will be pasted as $\text{Real.i} = \text{Diameter} + 10$



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

Specifying the Material Parameter



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

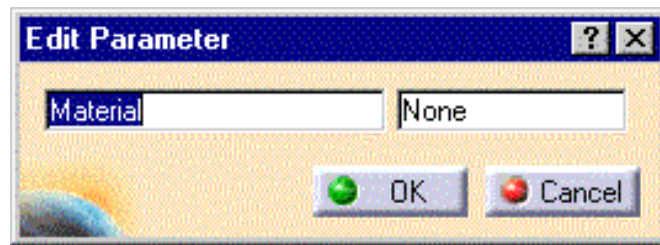


1. Open the [KwrStartDocument.CATPart](#) document.


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



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



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

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

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

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



Remember: To display parameter values, check

Tools->Options->General->Knowledge->Parameters and **Measure->With value**.

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



Valuating the Mechanical Property Parameters



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



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



Specifying a Parameter Value as a Measure




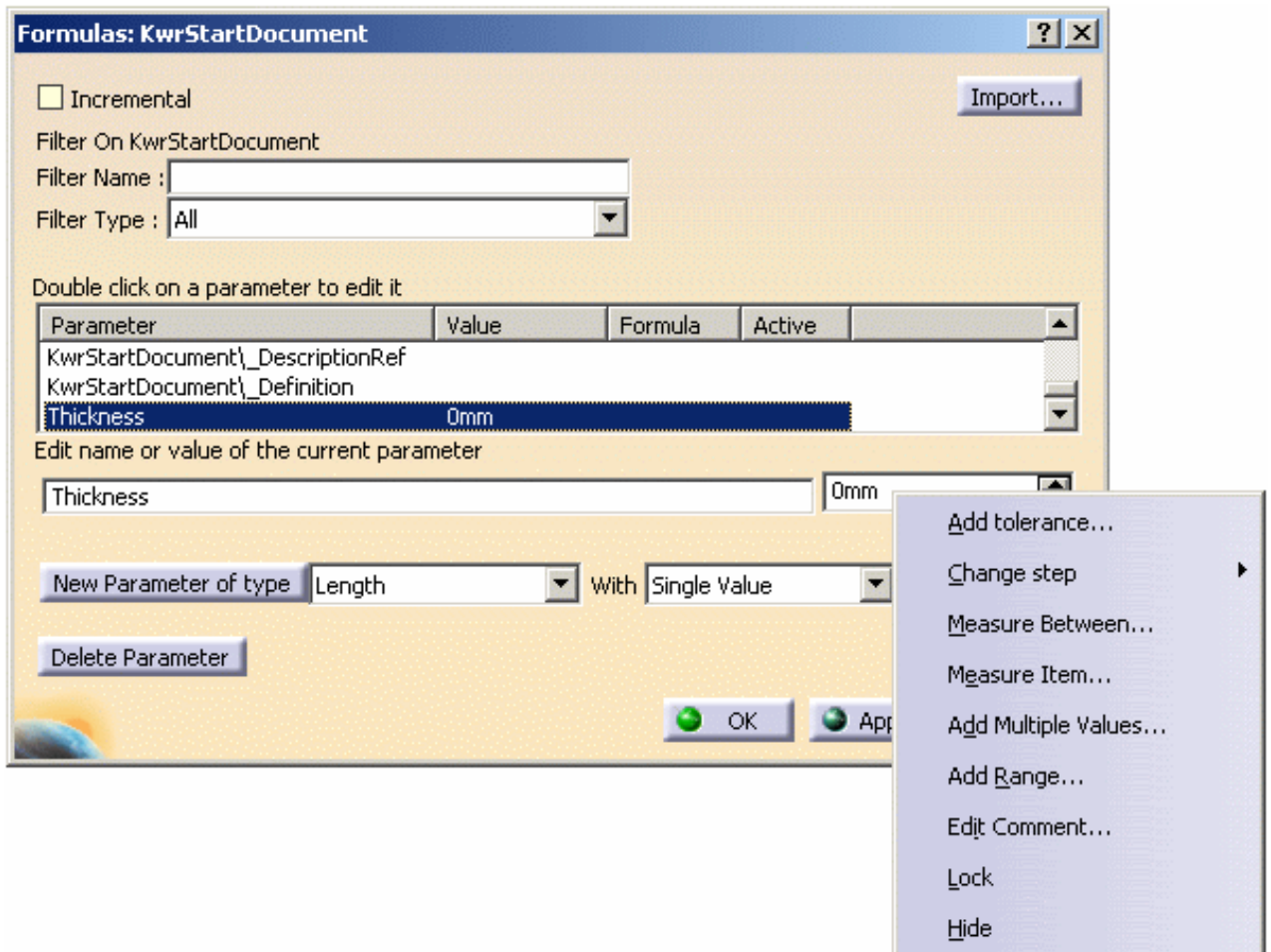
This scenario shows how to assign a value to a parameter deducing it from a graphic selection. In this scenario, the user deduces the value assigned to the Thickness parameter by selecting 2 circular edges.



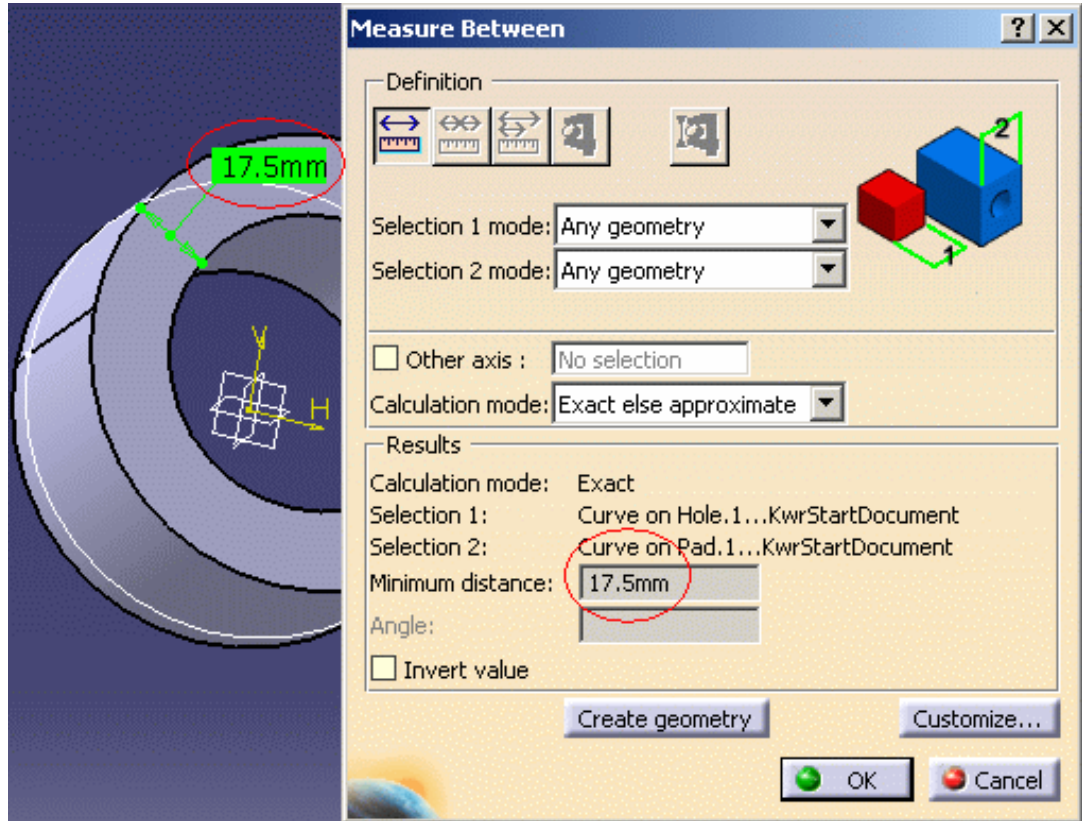
A common way to assign a value to a parameter is to use the Edit name or value of the current parameter field of the Formulas dialog box. But there is another way to proceed. The value you assign to a parameter can be deduced from a graphic selection.



1. In **Tools->Options->General->Parameters and Measure**, check the **Load extended language libraries** box of the Language tab.
2. Open the [KwrStartDocument.CATPart](#) document.
3. Click the  icon. The *f(x)* dialog box is displayed.
4. Select the **Length** item with **Single Value** in the **New Parameter of type** list, then click **New Parameter of type**.
The new parameter appears in Edit name or value of the current parameter.
5. Replace the Length.1 name with Thickness, then right-click in the value field of **Edit name or value of the current parameter**.




6. Select the **Measure Between...** command from the contextual menu. The Measure Between dialog box is displayed. Select Edge only as **Selection 1** mode and Edge only as **Selection 2** mode.
7. In the document geometry area, select successively one of the inner circular edge of the part, then the outer circular edge located on the same face. The 17.5 mm value is displayed in the Measure Between dialog box.

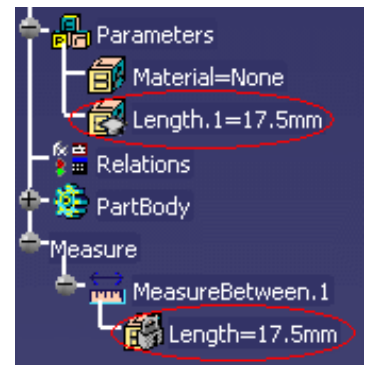


8. Click **OK** when done in the Measure Between dialog box. The 17.5 mm value is displays in the Formulas dialog box.
9. Click **OK** to close the Formulas dialog box.

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

To edit this parameter, proceed as follows:

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



Importing Parameters



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




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



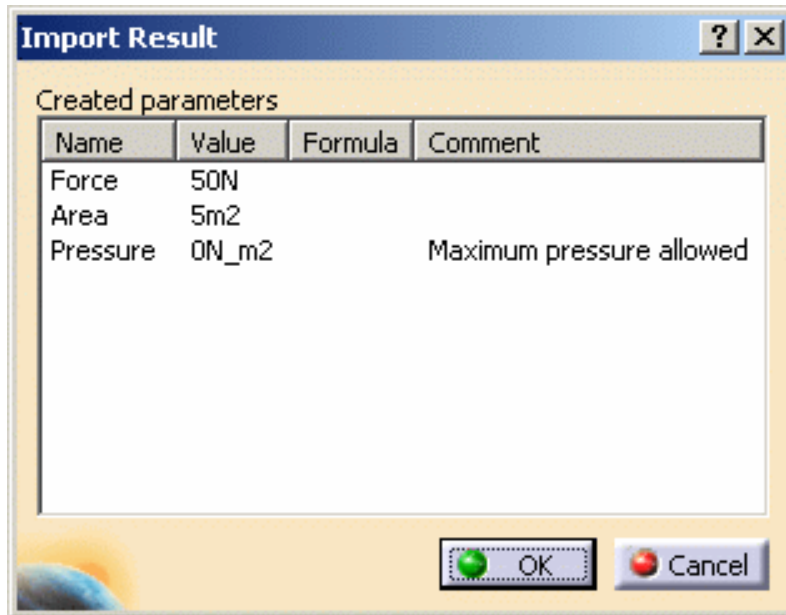
Please find below the formatting rules the external file should comply with:

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



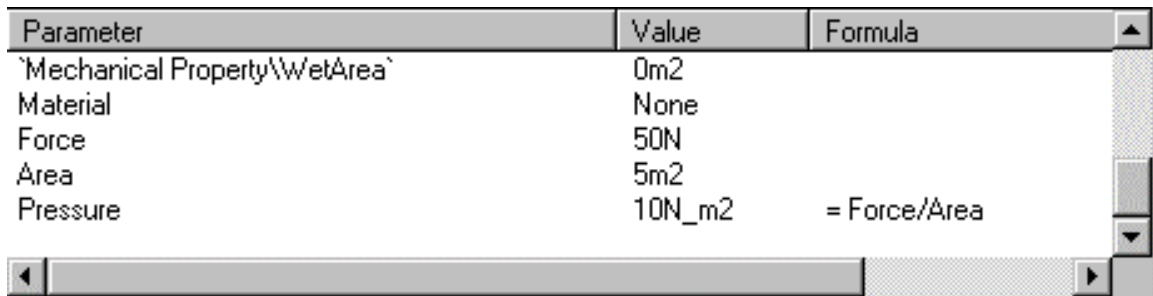
1. Open the [KwrStartDocument.CATPart](#) document.
2. Click the  icon. The *f(x)* dialog box is displayed.
3. Click Import.... A file selection dialog box is displayed.
4. Select the [ExCompanyFile0.xls](#) file (Windows only) or the [TxCompanyFile0.txt](#) file, then click **Open**.

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



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

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




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



Creating Points, Lines... as Parameters



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


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



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

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




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

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

The inertiaAxis function is accessible through the Line constructors. The axis number 3 is the one which is in the extrusion direction (normal to yz). Click OK in

the Formulas dialog box. The inertia axis is displayed in the geometry area.

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

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

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

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



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

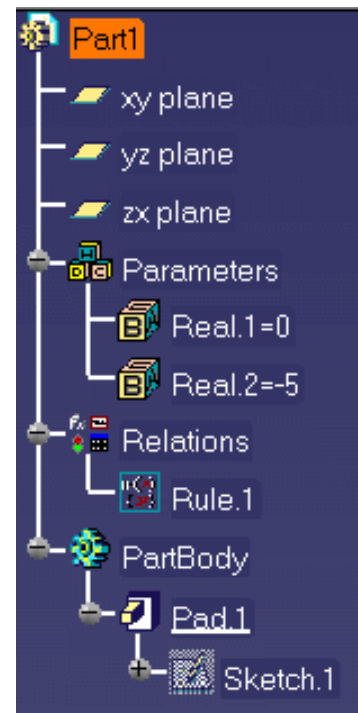


Applying Ranges to Parameters by Using a Rule

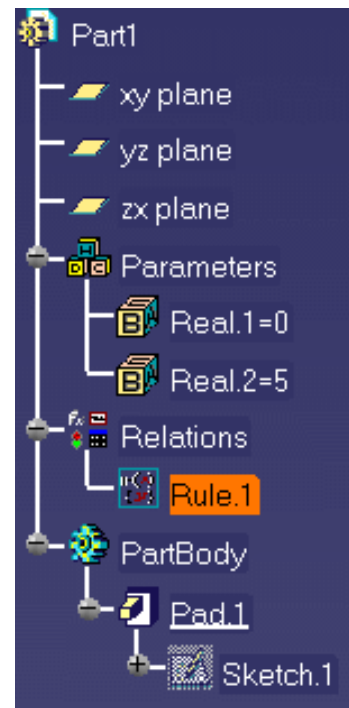


This task explains how to apply ranges to parameters by using a rule.

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



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



Activating and Deactivating a Component



This task explains how to activate and deactivate a component.


In the scenario described below, the CATProduct file contains two CATPart files that you will activate and deactivate alternatively after creating user parameters and a rule based on these parameters.



Parameters driven by rules are designed to enable the user to control components activities at assembly level.

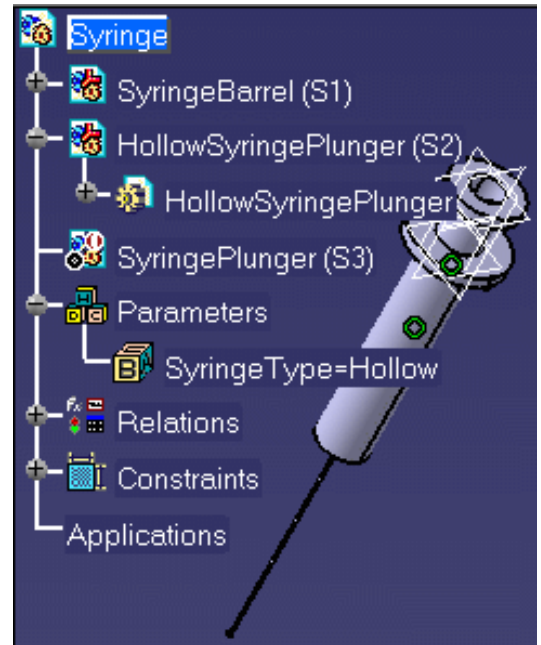


1. Open the [KwrSyringe.CATProduct](#) file and save the following files in the same directory ([SyringePiston.CATPart](#), [HollowSyringePiston.CATPart](#), and [SyringeContainer.CATPart](#)): This file contains a syringe made up of three different parts: A barrel, and two different plungers.
2. Create a multiple value parameters of string type. To do so, proceed as follows:

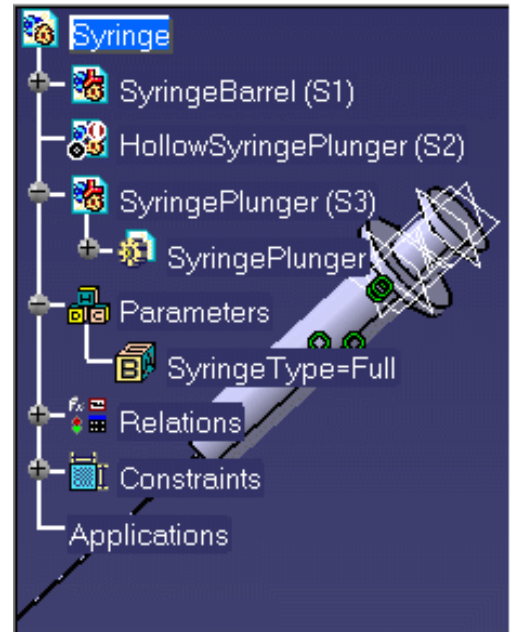
- Click the  icon. The Formulas Editor opens.
 - Select String in the scrolling list with Multiple Values. Click the **New Parameter of type** button. The **Value List** dialog box opens.
 - Enter two different values, Hollow and Full, and click **OK**.
 - Edit the name of the new parameter (SyringeType in this scenario) in the **Edit Name or value of the current parameter** and click **OK**. The new parameter is displayed under the **Parameters** node of the Specification tree.
3. Access the Knowledge Advisor workbench and click the Rule icon to create a rule. The script of this rule will allow you to enable or disable one of the plungers.
 4. Enter the code below in the Rule Editor, and click **OK**.

```
if (SyringeType == "Hollow")
{
S3\Component_Activation_State = false
S2\Component_Activation_State = true
}
else
{
S2\Component_Activation_State = false
S3\Component_Activation_State = true
}
```

6. Double-click the SyringeType parameter under the Parameters node and select Hollow in the Edit Parameter window. The SyringeBarrel CATPart and the HollowSyringePlunger CATPart are displayed.



7. Double-click the SyringeType parameter and select "Full" in the Edit Parameter window. The SyringeBarrel CATPart and the SyringePlunger CATPart are displayed.



Creating an Associative Link Between Measures and Parameters



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

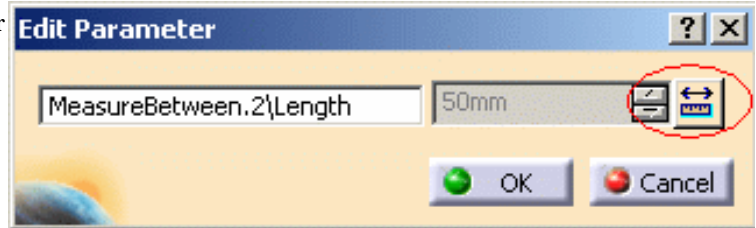
- **Measure Item** allows you to get the length of a curve (edge, line, curve), radius or angle depending on the parameter magnitude.
- **Measure Between** allows you to get the minimal distance or angle between two elements, depending on the parameter magnitude.



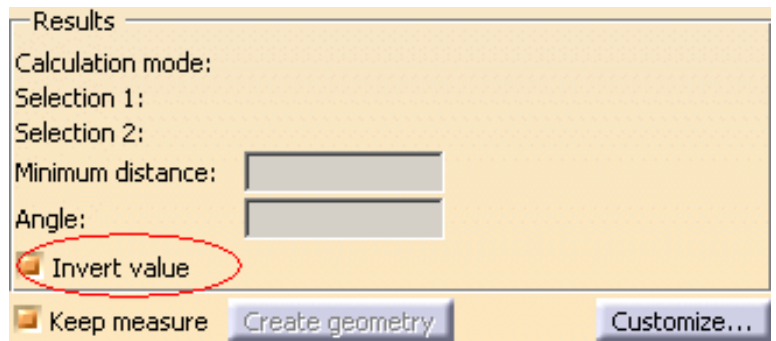
This link can be created only if the **Keep measure** option is checked in the **Measure Item** and **Measure Between** dialog boxes (if not the result is copied as a simple value.)



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



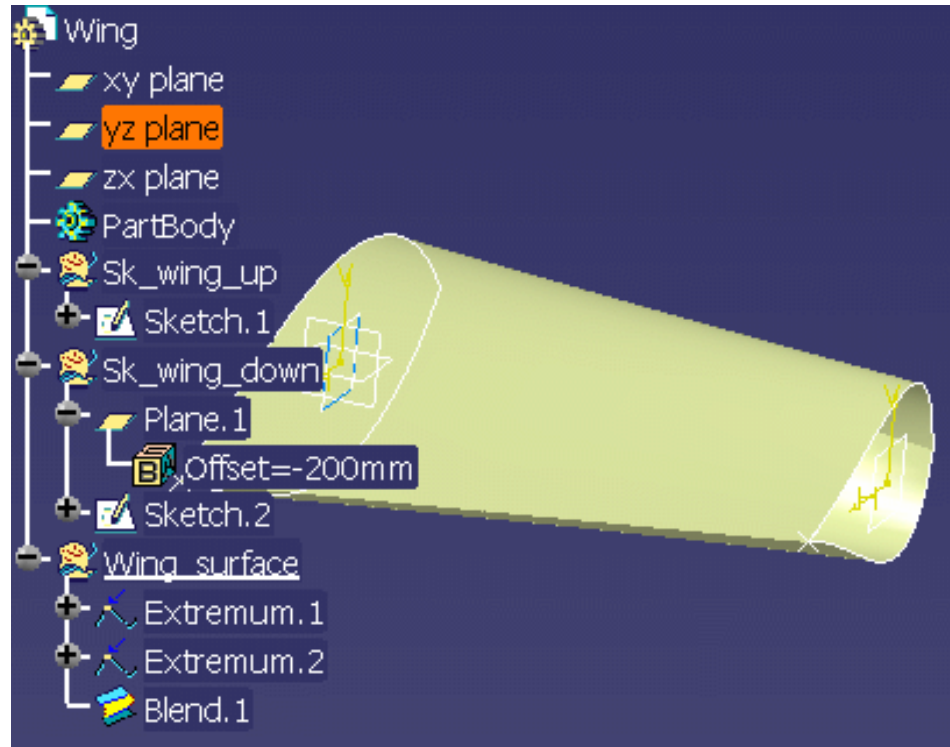
- The parameters located below the Parameters node are directly linked to the measures.
- You can invert the sign of the parameter using the **Invert value** command in the Measure Item or Measure Between panel. The sign concerns only the valuated parameters and not the parameter of the measure.



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



1. Open the [KwrPlaneWing.CATPart](#) file. The following image displays.

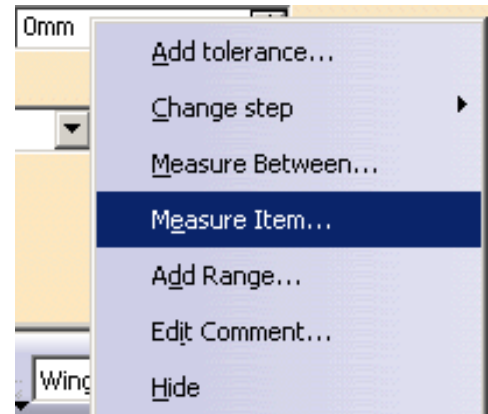


Using the Measure Item... command

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

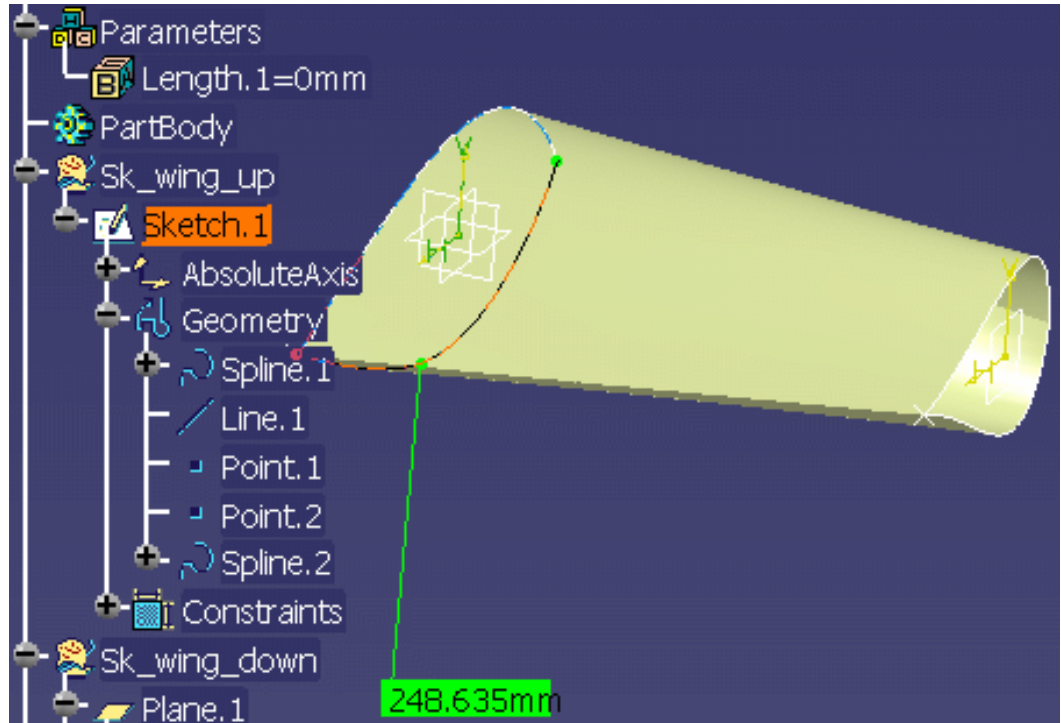
- Click the Formula icon (). The Formulas dialog box displays.
- In the **New parameter of type** scrolling list, select **Length** and click the **New parameter of type** button. **Length.1** displays in the **Edit name or value of the current parameter** field.

- Right-click the value field of **Length.1** and select the **Measure Item...** command. The **Measure Item** dialog box displays.



- Make sure the **Keep measure** option is checked in the **Measure Item** dialog box.

- In the specification tree, expand the Sketch.1 node, and select Spline.2. The selected item is highlighted in the geometry and its measure is displayed in green.




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

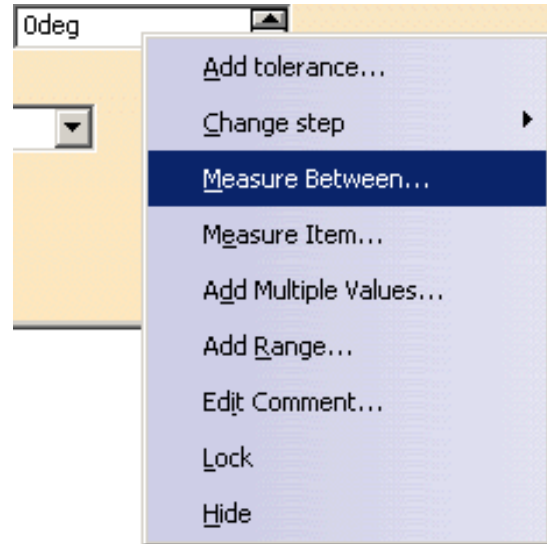


Using the Measure Between... command

1. Add a parameter of **Angle** type. To do so, proceed as follows:

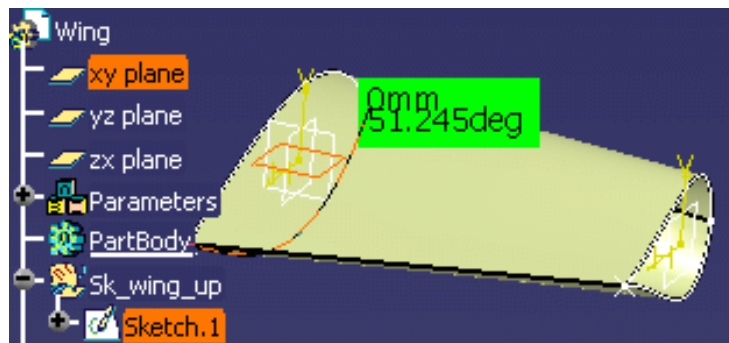
- Click the Formula icon (). The Formulas dialog box displays.
- In the **New parameter of type** scrolling list, select **Angle** and click the **New parameter of type** button. **Angle.1** displays in the **Edit name or value of the current parameter** field.

- Right-click the value field of **Angle.1** and select the **Measure Between...** command. The **Measure Between** dialog box displays.

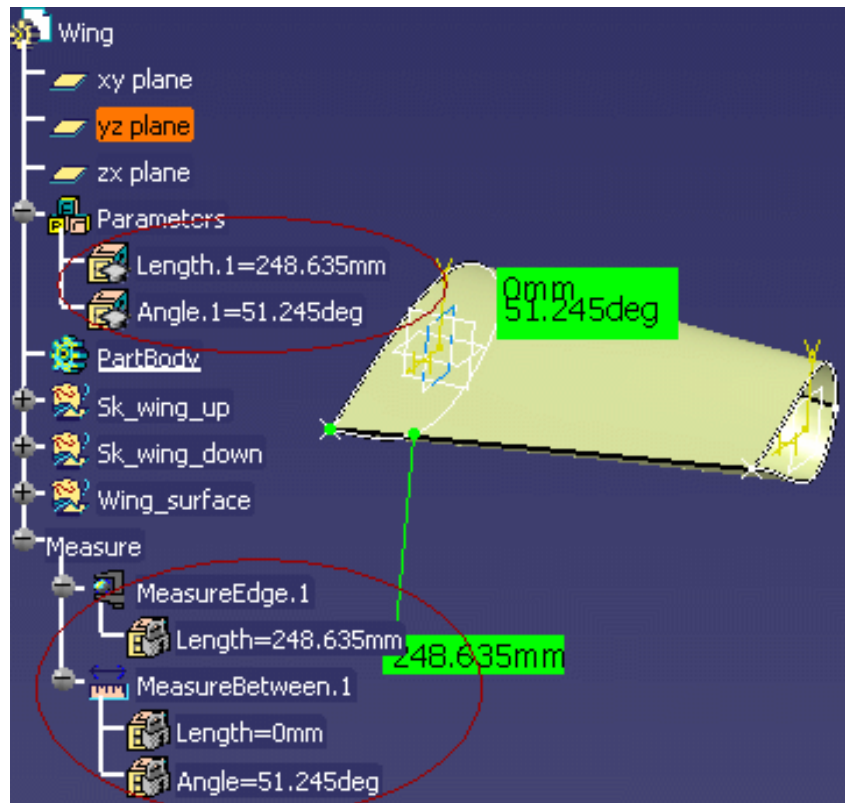


- In the **Selection 2 mode** scrolling list, select the **Edge only** option.
- In the specification tree, select Plane xy then select the geometry as shown below.

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



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



○ Click [here](#) to display the result sample.



Note that:

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

Using Relations based on Publications at the Product Level



This scenario explains how to use relations based on publications at the product level. The scenario described below is divided into the following steps:

- Add a parameter to the KwrScrew.CATPart called Screw_Volume, add a formula to calculate the volume of the screw and publish the Screw_Volume parameter.
- Add a parameter to the KwrScrew1.CATPart called Screw_Volume, add a formula to calculate the volume of the screw and publish the Screw_Volume parameter.
- Create a CATProduct file called Bolt and import the KwrScrew.CATPart
- Import KwrNut.CATPart.
- In the context of the Bolt product, create a formula calculating the bolt volume based on the screw and the nut publications.
- In the context of the bolt, replace KwrScrew.CATPart by KwrScrew1.CATPart. The volume is recomputed.



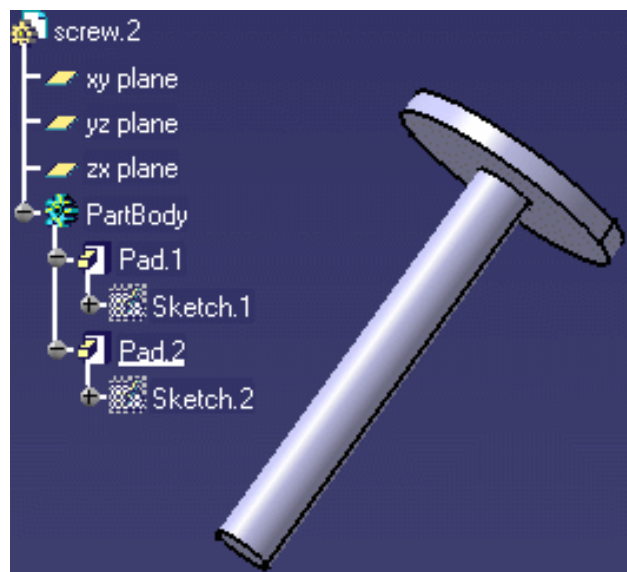
Before you start, make sure that the **Keep link with selected object** check box is checked (**Tools->Options->Infrastructure->Part Infrastructure->General**).




Note that this function can be used with:

- Design Tables
- Formulas
- Rules and Checks
- Set of Equations
- The optimization

1. Open the [KwrNewScrew.CATPart](#) document. The following image displays.



2. Add a Volume parameter to the part. To do so, proceed as follows:

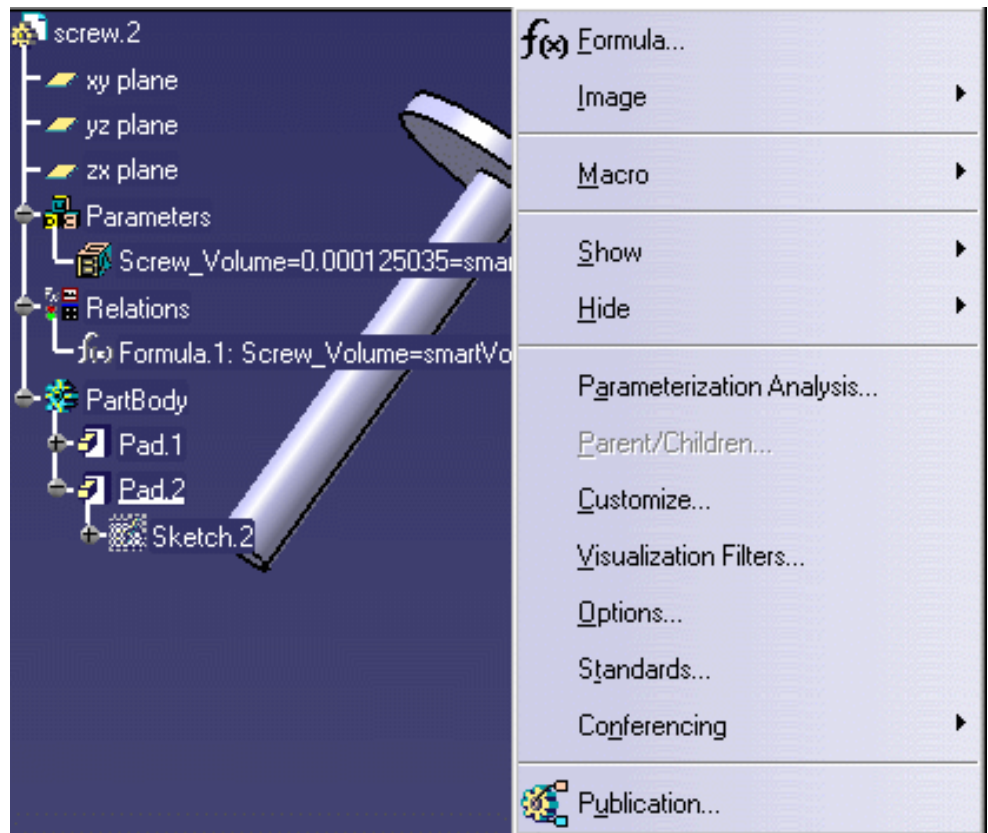
- Click the  icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Real** and click the **New parameter of type** button.
- In the **Edit name or value of the current parameter** field, enter the name of the parameter: **Screw_Volume**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.
- Enter the following formula by using the Dictionary:

Screw_Volume=smartVolume(PartBody\Pad.1)+smartVolume(PartBody\Pad.2).

Click **OK** three times.

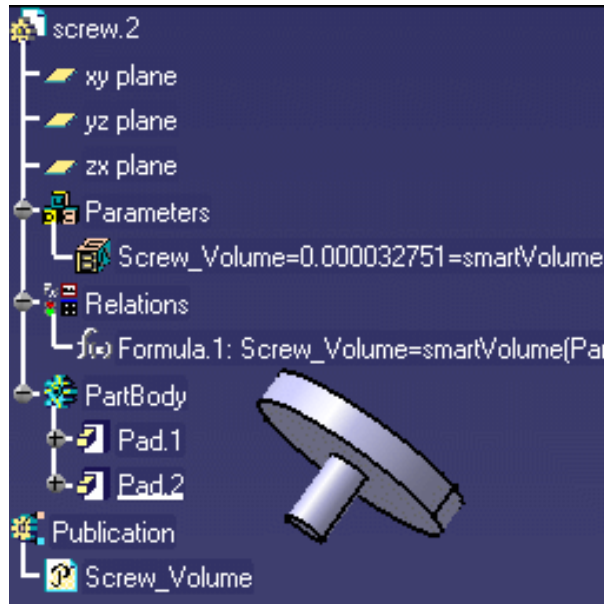
3. Publish the **Screw_Volume** parameter.

To do so, select the **Tools->Publication** command and click the **Screw_Volume** parameter under the Parameters node in the specification tree. Click **OK**. The published parameter appears in the specifications tree below the Publication node. Save your file and close it.

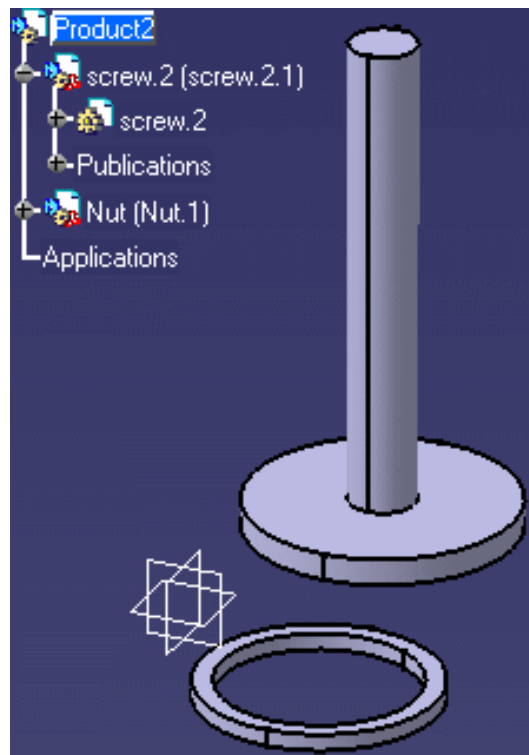


4. Open the [KwrNewScrew1.CATPart](#) and repeat the steps listed above (steps 1 to 3 included). The


part should be identical to the one below. Save your file and close it.

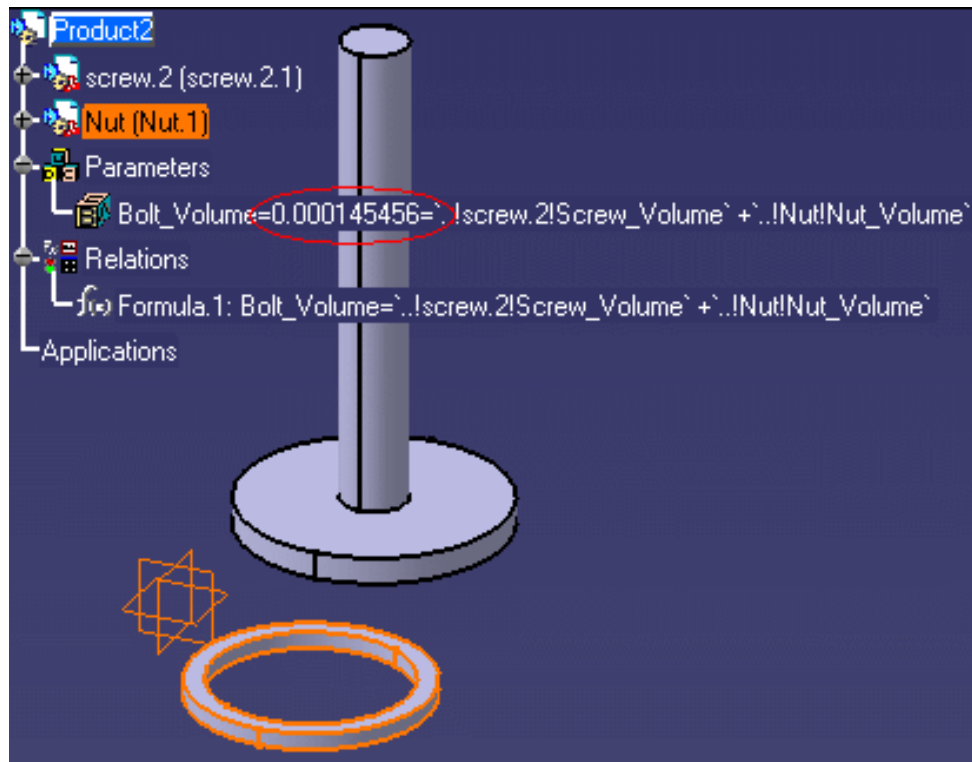


5. Create a CATProduct file named KwrBolt.CATProduct.
6. Click the Root product and select the **Insert->Existing Component...** command. The **File selection** box displays. Select the KwrNewScrew.CATPart file and click **Open**. The screw is imported.
7. Select the **Insert->Existing Component...** command, select the [KwrNewnut.CATPart](#) file and click **Open**. The nut part is inserted.

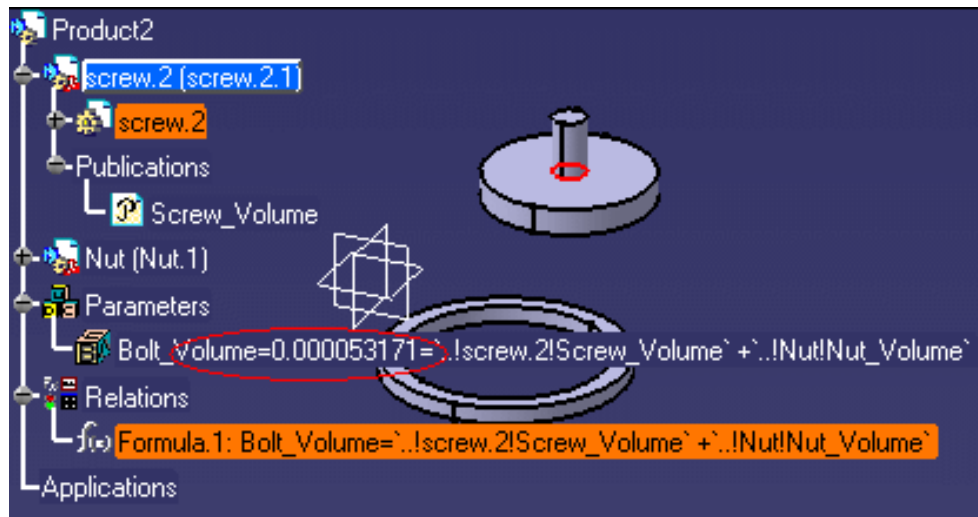


8. Add a Bolt_Volume parameter to the product to compute the volume of the bolt. To do so, proceed as follows:

- o Click the Root product and click the  icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Real** and click the **New parameter of type** button.
- o In the **Edit name or value of the current parameter** field, enter the name of the parameter: **Bolt_Volume**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.
- o Enter the following formula by using the Dictionary and by clicking the published parameters in the specification tree: **Bolt_Volume = ` ..!screw.2!Screw_Volume` + ` ..!Nut!Nut_Volume`**. Click **OK**, and **OK**. The Bolt volume displays



9. Replace the screw to compute a new volume: Double-click, then right-click the Screw.2 component in the specification tree and select the **Components->Replace Component...** command. The **File Selection** window opens. Select the KwrNewScrew1.CATPart file and click **Open**.
10. Click **Yes** and **OK** in the Impacts on Replace window. The new screw is inserted and the bolt volume is updated.



Publishing Parameters



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

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



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

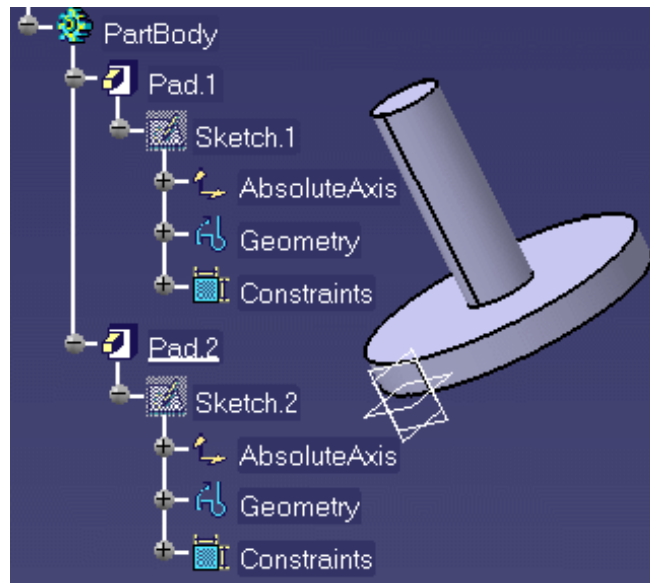
The publication of parameters should be used when:

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




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

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



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

- Click the  icon. The Formula Editor opens. In the **New parameter of type** scrolling list, select **Volume** and click the **New parameter of type** button.
- In the **Edit name or value of the current parameter** field, enter the name of the parameter: **MyVolume**. Click **Apply** and click the **Add Formula** button. The Formula Editor opens.

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

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

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

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

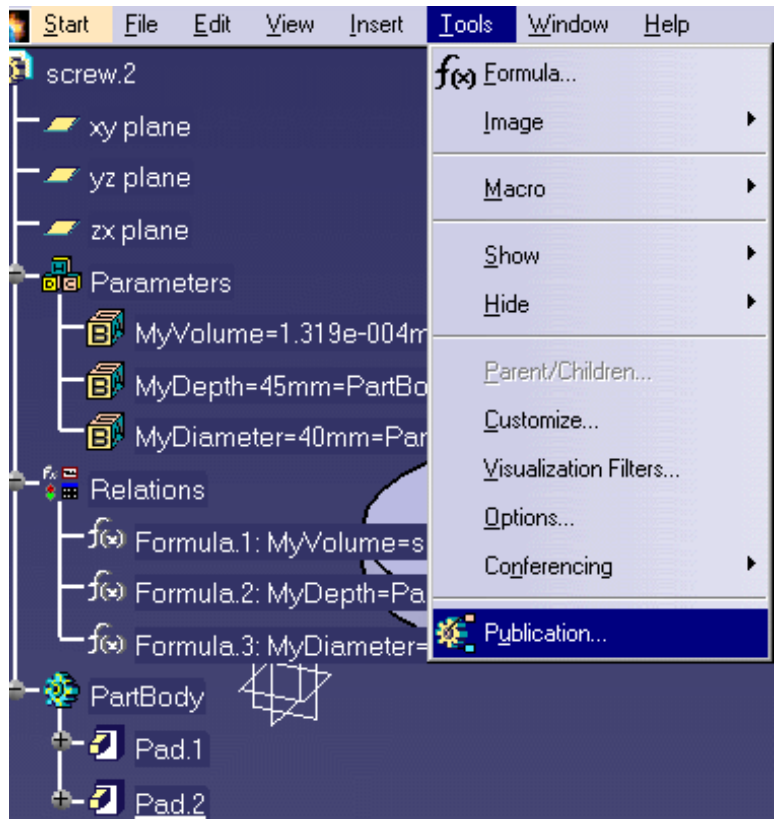
- o In the **New parameter of type** scrolling list, select **Length** and click the **New parameter of type** button.

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

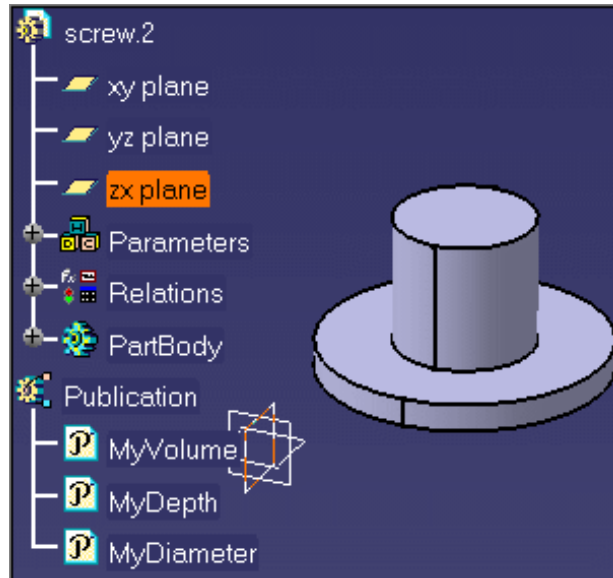
- o Enter the following formula: **MyDiameter=PartBody\Sketch.2\Radius.2\Radius * 2**. Click **OK** twice.

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

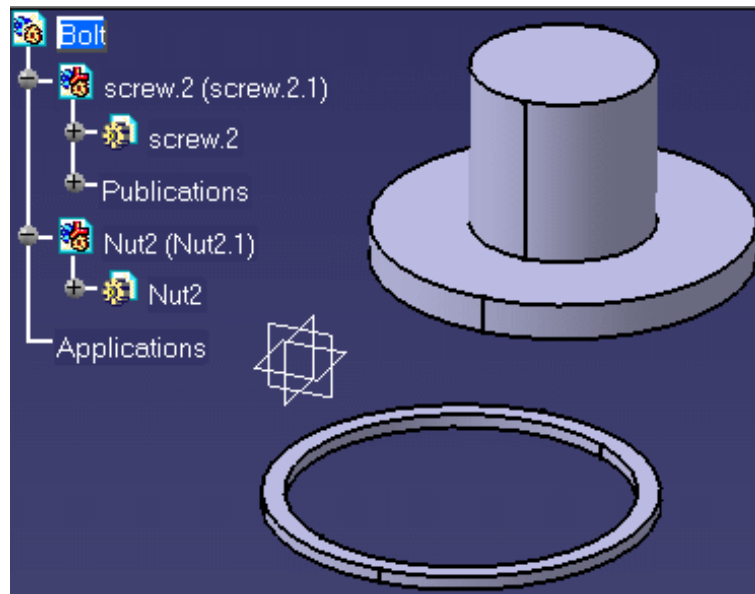
To do so, select the **Tools->Publication** command and select the **MyVolume**, **MyDepth**, and **MyDiameter** parameters in the specifications tree. Click **OK**. The published parameters appear in the specifications tree below the Publication node. Close the file.



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



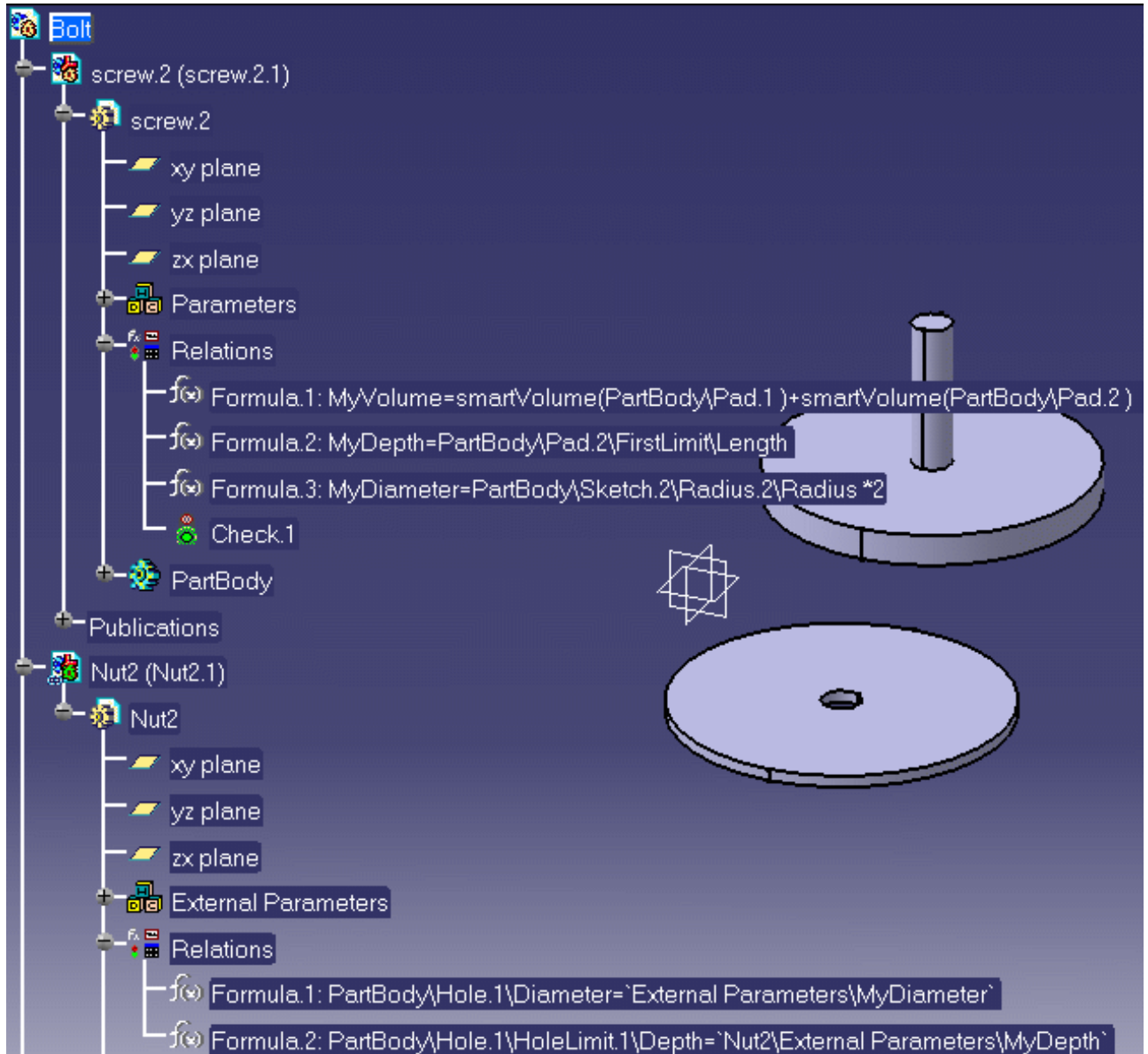
5. Create a CATProduct file. Select the **Insert->Existing Component...** command and click the root of the specifications tree. The **File selection** box displays. Select the [KwrScrew.CATPart](#) file and click **Open**. The screw is imported.
6. Select the **Insert->Existing Component...** command, select the [Kwrnut.CATPart](#) file and click **Open**. The nut part is inserted.



7. Double-click the inner circle of the nut, the Hole Definition window displays.
 - o Right-click the **Diameter** field and select the **Edit formula...** command. The Formula Editor opens.
 - o Select **MyDiameter** in the screw publications. The formula should be as follows:
PartBody\Hole.1\Diameter= `External Parameters\MyDiameter`. Click **OK**.

- o Right-click the **Depth** field and select the **Edit formula...** command. The Formula Editor opens.
- o Select MyDepth in the screw publications. The formula should be as follows:
PartBody\Hole.1\HoleLimit.1\Depth= `External Parameters\MyDepth`. Click **OK** twice.

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



Parameters: Useful Tips

- You can add properties to a .CATPart or a .CATProduct document by using the Properties command from the contextual menu. You just have to click the Define other properties... button in the Product tab then click New parameter of type. The dialog is similar to the $f(x)$ dialog. See the *Product Structure User's Guide* for more information. The properties you define that way are also displayed in the parameter list of the $f(x)$ dialog box.
- You can specify that a parameter is constant by using the Properties command from the contextual menu. This command also enables you to hide a parameter.
- When copying parameters sets containing hidden parameters, these parameters are automatically pasted when pasting the parameters sets and appear as hidden parameters.
- Parameters have 2 different names: The **local** one and the **global** one.
 - The local name is the name attributed to the parameter when it was created in the Formula Editor or in the Parameters Explorer. Note that this name will not be modified if you perform a Reorder using the contextual menu. The local name can only be modified using the Parameters Explorer.
 - The global name (name) is the name attributed to the parameter by Knowledge Advisor. It is the path of the parameter + its type. If you select the parameter and reorder it, the path contained in the name will be modified. If you double-click the parameter in the specification tree, and enter a new name in the Edit Parameter dialog box, the global name will be changed. If, after renaming the parameter in the Edit Parameter dialog box, you reorder the parameter the path will not appear any more.

Formulas

Introducing Formulas

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

Using the Dictionary

Creating a Formula

Creating Formulas based on Publications

Specifying a Measure in a Formula

Referring to External Parameters in a Formula

Using the Equivalent Dimensions Feature

Creating Formulas based on Publications

Getting Familiar with the Equivalent Dimensions Interface

Controlling Relations Updates

Formulas: Useful Tips



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

Introducing Formulas

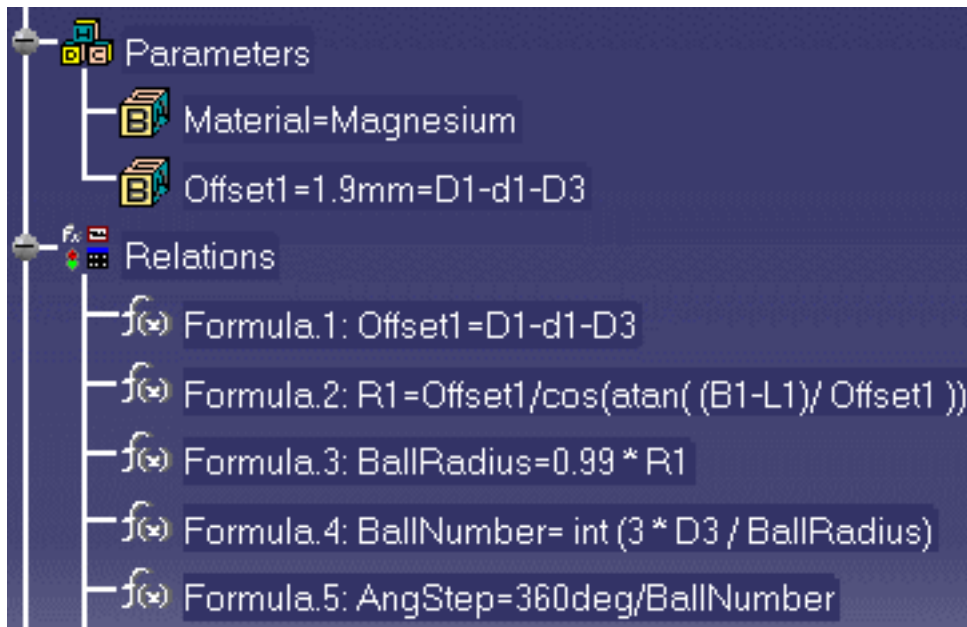
Formulas are features used to define or constrain a parameter. A formula is a relation: the left part of the relation is the parameter to be constrained, the right part is a statement. Once it has been created, a formula can be manipulated like any other feature from its contextual menu. The formula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

Displaying Formulas in the Specification Tree

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

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

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



The Activity Parameter

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

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

parameter.

Activity value

**Relation icon in the
specification tree**

false




true



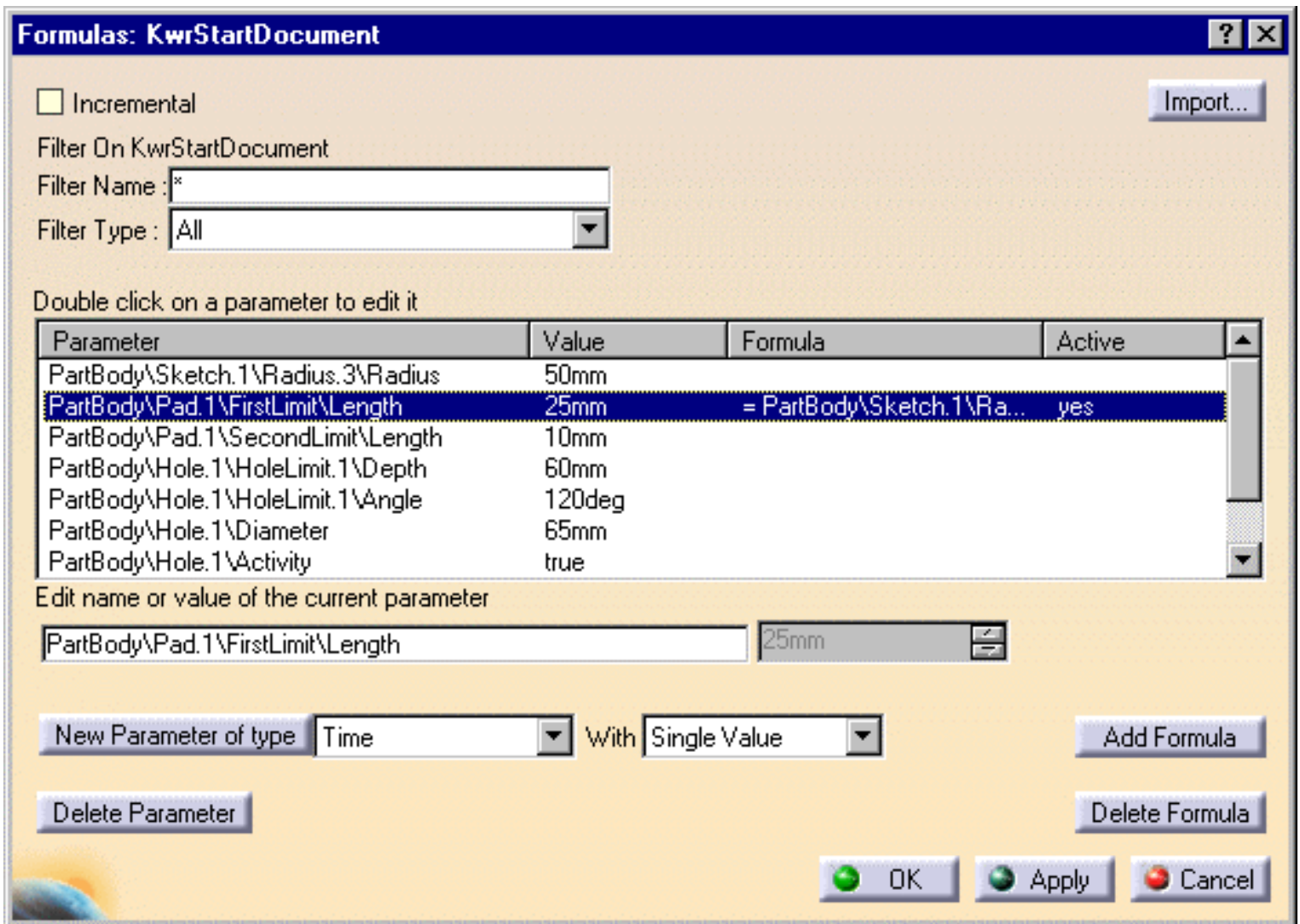
Importing Formulas

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

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

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

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



The parameter list

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

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

as the Incremental check box.

The Filter Name filter

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

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

The Filter Type filter

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



The Hide command is only available for user parameters.

The Incremental check box

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

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

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

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

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

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

The New Parameter of type button

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

The Delete Parameter button

This capability operates only for user parameters.

The Add Formula button

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

To know more about the Dictionary available in the Formula editor, see [Using the Dictionary](#).

The Delete Formula button

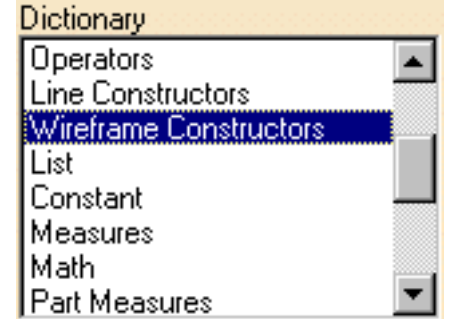
When a parameter which is constrained by a formula is selected in the parameter list, clicking Delete Formula removes the formula.

The Import button

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

Using the Dictionary

To help you write a formula, the formula editor provides you with a dictionary. This dictionary exposes the list of parameters and functions you can use to define a formula. Depending on the category of objects to be referred to in the formula, the dictionary is divided into two or three parts. To insert any definition in the formula editor, just double-click the object either in the dictionary or in the tree. If you double-click a function in the dictionary, its signature is carried forward to the formula editor. Only the argument definitions are missing.



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

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

Design Table Methods

ClosersupConfig Method	ClosersupConfig Function	CloseValueSupInColumn Method
CloseValueInfInColumn Method	MinInColumn Function	MaxInColumn Method
LocateInColumn Method	CellAsString Function	CellAsBoolean Method
CellAsReal Method	SetCell Method	LocateInRow Method

ClosersupConfig Method

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

Syntax

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

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

Arguments	Description
<i>columnName</i>	Should be put in quotes. At least, one couple of arguments <i>columnName_i/minValue_i</i> is required
<i>minValue</i>	Required. You should specify the units.

Example

Given the design table below:

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

The expression below:

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

returns 3

ClosersupConfig Method

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

Syntax

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

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

Arguments	Description
<i>columnName</i>	Should be put in quotes. At least, one couple <i>columnName/maxValue</i> is required

<i>maxValue</i>	Required. You should specify the units.
-----------------	-----------------------------------------

Example

Given the design table below:

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

The statement below

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

Explanations

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

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

CloserValueSupInColumn Method

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

Syntax

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

The **CloserValueSupInColumn** method takes two arguments:

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

Example

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

CloserValueInfInColumn Method

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

Syntax

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

The **CloserValueInfInColumn** function has two arguments:

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

Example

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

MinInColumn Method

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

Syntax

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

where *columnIndex* is the column number.

Example

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

Sample

[KwrProgramDT.CATPart](#)

MaxInColumn Method

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

Syntax

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

Example

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

LocateInColumn Method

Applies to a design table sheet. Returns the index of the first row which contains a specified value. Returns zero if the value is not

found or if the method arguments are not properly specified.

Syntax

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

The **LocateInColumn** method has two arguments:

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

Example

```
Line= Relations\DesignTable1\sheet_name.LocateInColumn(4, 1 1 mm)
if (Line == 0)
{
  Message("No value found !!!")
}
```

CellAsString Method

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

Syntax

sheet.**CellAsString**(*rowIndex*: Integer, *columnIndex*: Integer): String

where *rowIndex* is the configuration number and *columnIndex* the column number.

Example

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

CellAsBoolean Method

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

Syntax

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

The **CellAsBoolean** method has two arguments:

Arguments	Description
<i>rowIndex</i>	Required. Configuration number. Integer from 1 to n.
<i>columnIndex</i>	Required. Index of the table column. Integer from 1 to n.

Example

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

CellAsReal Method

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

Syntax

sheet.**CellAsReal**(*RowIndex*: Integer, *ColumnIndex*: Integer): Real

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

SetCell Method

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

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

Syntax

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

LocateInRow

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

Syntax

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

The **LocateInRow** method has two arguments:

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

Operators

Arithmetic operators

- + Addition operator (also concatenates strings)
- Subtraction operator
- * Multiplication operator
- / Division operator
- () Parentheses (used to group operands in expressions)
- = Assignment operator
- ** Exponentiation operator

Logical Operators

- and** Logical conjunction on two expressions
- or** Logical disjunction on two expressions

Comparison Operators

- <> Not equal to
- == Equal to
- >= Greater or equal to
- <= Less than or equal to
- < Less than
- > Greater than

Point Constructors

Sample: [KwrPointConstructors](#)

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

Examples:

Specifying values:

```
Geometrical Set.1\Point.1 =
point(10mm,10mm,10mm)
```

Specifying parameter names:

```
Geometrical Set.1\Point.4 =
point(0mm,L3,L1)
```

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

Geometrical Set.1\Point.2=

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

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

Evaluate Method


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

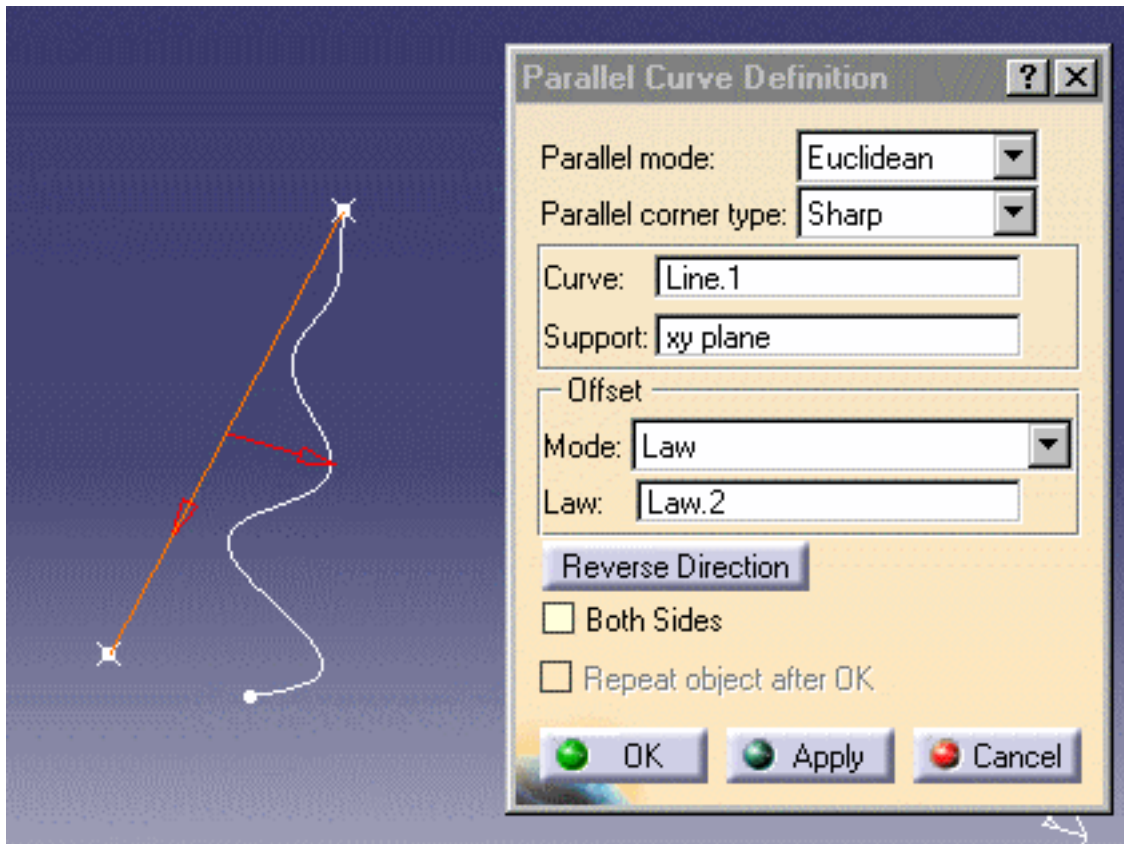
Syntax

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

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

Example

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



Sample

KwrObject.CATPart

Line Constructors

Sample: [KwrLineConstructors](#)

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

Example:

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

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

Example:

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

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

Example:

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

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

Example:

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

- **linenormal**(*sur: Surface, pt:Point, start:Length, end:Length, orientation: Boolean*) : Line
Creates a line normal to a surface at a given point.
- **mainnormal**(*crv: Curve, pt: Point*) : Line
Creates a line normal to a curve at a given point.
The line is created in the plane which contains the tangent vector.
- **binormal**(*crv: Curve, pt: Point*) : Line
Creates a line normal to a curve at a given point.
The line is created in plane which is orthogonal to the tangent vector.

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

Example:

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

Circle Constructors

Sample: [KwrCircleConstructors.CATPart](#)

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

Creates a circular arc from its center and radius. If the argument 4 is 0, the arguments 5 and 6 are taken into account. Otherwise, a circle is created.

Example

```
Geometrical Set.1\Circle.1 =
circleCtrRadius(Geometrical Set.1\Point.1 , `zx plane` ,20mm,0,10deg,320deg)
```

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

Creates a circular arc from its center and another point located on the circle. If the argument 4 is 0, the arguments 5 and 6 are taken into account. Otherwise, a circle is created.

Example

```
Geometrical Set.1\Circle.2 =
circleCtrPt(Geometrical Set.1\Point.1 , Geometrical Set.1\Point.2 , `xy plane` ,1,10deg, 370deg)
```

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

Creates a circular arc. The points specified in the arguments 1 and 2 are located on the arc to be created and define the arc limits when the integer specified in the argument 6 is 0. When 0 is specified in the argument 6, modifying the argument 5 boolean value allows you to display the alternative arc.

Example

```
Geometrical Set.1\Circle.3 =
circle2PtsRadius(Geometrical Set.1\Point.1 ,Geometrical Set.1\Point.2 , `xy plane` ,50mm, true, 0)
```

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

Creates one or more circular arcs passing through three points. When 0 is specified in the argument 4, the first and third points define the arc limits. When 1 is specified in the argument 4 the whole circle is defined. When 2 is specifies in the argument 4 the direct circle is defined. When 3 is specified in the argument 4, the complementary circle is defined.

Example

```
Geometrical Set.1\Circle.2 =
circle3Pts(Geometrical Set.1\Point.1, Geometrical Set.1\Point.2, Geometrical Set.1\Point.3, 0)
```

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

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

Example

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

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

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

Example

Geometrical Set.1\Circle.4 =

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

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

Creates one or more circular arcs tangent to two curves.

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

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

Example

Geometrical Set.1\Circle.6 =

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

List

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

- **List.Size ()** : Integer
Method used to return the number of items contained in the list.
- **List.AddItem** (*Object: Objecttype, Index: Integer*):VoidType
Method used to add an item to the list.

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

- **List.RemoveItem** (*Index: Integer*) :VoidType
Method used to remove an item from the list.
- **List.GetItem** (*Index: Integer*) :ObjectType
Method used to retrieve a value/item from the list
- **List.ReorderItem** (*Current: Integer, Target: Integer*) :ObjectType
Method used to move an element of the list to a new position.
- **Copy** (List: List) : List
Method used to copy the content of a list and paste it in another list.
- **List** (Next: ObjectType, ...): List
Method used to create a list.
- **List.Sum** (): Real
Computes the sum of the items contained in the list..
- **List.IndexOf** (Element: ObjectType, StartIndex:Integer):Integer
Returns the index of a list item.

- **Compute()**

Function used to compute the result of an operation performed on the attributes supported by the features contained in the list.

Example: List.1 .Compute("+", "Hole", "x.Diameter", Length.1)

Where:

- List.1 is the name of the list on which the calculation will be performed
- + is the operator used. (Supported operators are: -, min, and max.)
- Hole is the type of the list items used for the calculation (to calculate the diameter, the type to be indicated is Hole, to calculate the volume, the type to be indicated is Solid)
- x stands for the list items. Note that the type of the items contained in the list should be identical.
- Length.1 is the output parameter.

Measures

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

Sample: [KwrMeasuresWiz.CATPart](#)

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

Example:

Length.1 = distance(Body.3 , Body.1)

- **minimumCurvatureRadius(Curve):Length**
For an item of dimension 1 (a curve), enables the user to measure its minimum radius of curvature.
- **nbDomains(Body): Integer**
For all types of items, enables the user to compute the number of domains.
- **length(GSMCurve) : Length**
Returns the total length of a curve.
- **length(GSMCurve, Point1, Point2) : Length**
Returns the length of a curve segment delimited by *Point1* and *Point2*.
- **length(GSMCurve, Point1, Boolean): Length**
Returns the length of a curve segment located between *Point1* and one of the curve ends. Modifying the boolean value allows you to retrieve the length from the specified point to the other end.
- **area(Surface): Area**
Returns the area of a surface generated by the Generative Shape Design product (an extruded surface for example).
- **area(Curve) : Area**
Returns the area delimited by a curve.
- **point.coord(Integer): Length**
Returns the coordinate of a point. Returns X if 1 is specified, Y if 2 is specified, Z if 3 is specified.

- **point.coord**(*oX*: Length, *oY*: Length, *oZ*: length): Void
Assigns the point coordinates to the length parameters specified in the arguments. This method can only be used in Knowledge Advisor rules.
- Example:
if Geometrical Set.1\Point.2.coord(1) > 0mm
Message("Point.2 abscissa is positive")
else
{
Geometrical Set.1\Point.1.coord(Xout, Yout, Zout)
Message("Point.1 abscissa is: # ", Xout)
}
- **volume**(*closedSurface*) : Volume
Returns the volume of a closed surface.
- **angle**(*Center: Point1, Pt1: Point2, Pt2*) : Angle
Returns the angle between the lines "C-Point1" and "C-Point2".
- **angle**(*Direction, Direction*) : Angle
Returns the angle between two directions.
- **angle**(*Line, Line*) : Angle
Returns the angle between the *Line1* and *Line2* lines.
- **angle**(*Plane, Plane*) : Angle
Returns the angle between t2 planes.
- **angleoriented(Direction, Direction, Direction): Angle**
Returns the angle between 2 directions and oriented by a third direction.
- **angleoriented(Line, Line, Direction): Angle**
Returns an angle between 2 lines and oriented by the direction.
- **angleoriented(Plane, Plane, Direction): Angle**
Returns an angle between 2 planes and oriented by the direction.
- **curvature**(*crv*: Curve, *pt*: Point): Real
Returns the curvature of a curve in a given point.

Example:

```
Real.1=  
curvature(Geometrical Set.1\Spline.1 ,Geometrical Set.1\Point.2 )
```

Surface Constructors

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

Geometrical Set.1\Surface.2=

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

Wireframe Constructors

<code>spline(pt: Point, ...): Curve</code>	<code>intersect(crv: Curve, crv: Curve) : Point</code>	<code>intersect(crv: Curve, su: Surface) : Point</code>	<code>intersect(su: Surface, su: Surface) : Curve</code>
<code>curveparallel(crv: Curve, su: Surface, offset: Length) : Curve</code>	<code>project(topproject: Point, support: Curve): Point</code>	<code>project(topproject: Point, support: Surface): Point</code>	<code>project(topproject: Point, support: Surface): Surface</code>
<code>assemble(Curve,...):Curve</code>	<code>corner(crv1: Curve, crv2: Curve, support: Surface, radius: Length, orientationcrv1: Boolean, orientationcrv2: Boolean, trim: Boolean) : Curve</code>	<code>split(tosplit: curve, splitting: Wireframe, orientation: Boolean): Curve</code>	<code>trim(crv1: Curve, orientationCrv1: Boolean, orientationCrv1: Boolean, crv1: Curve, orientationCrv2, Boolean): Curve</code>
<code>near(crv: Curve, near: Wireframe): Curve</code>	<code>near(crv: Point, near: Wireframe): Point</code>	<code>extrude(Point, Direction, length1: Length, length2: Length, orientation: Boolean): Line</code>	<code>revolve(Point, axis, Line: angle1, Angle: angle2, Angle): Circle</code>

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example

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

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

Example:

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

- **project**(*toproject*: Point, *support*: Surface): Point
Projects a point on a surface.
- **project**(*toproject*: Point, *support*: Surface): Surface
Projects a curve on a surface.
- **assemble**(Curve,...): Curve
Creates a join of several curves.
- **corner**(*crv1*: Curve, *crv2*: Curve, *support*: Surface, *radius*: Length, *orientationcrv1*: Boolean, *orientationcrv2*: Boolean, *trim*: Boolean) : Curve
Constructs a corner between two curves. The arguments 5 and 6 should be used to scan the possible solutions. See the *Generative Shape Design User's Guide* for more information on corners.

Example

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

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

Example

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

- **trim**(crv1: Curve, orientationCrv1: Boolean, orientationCrv2: Boolean, crv2: Curve, orientationCrv2, Boolean): Curve
Enables to trim two wireframe elements.

Example

Geometrical Set.1\Curve.1=

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


- **near**(crv: Curve, near: Wireframe): Curve
Creates the nearest entity of several sub-element. The result is a curve.
- **near**(crv: Point, near: Wireframe): Point
Creates the nearest entity of several sub-element. The result is a point.
- **extrude**(Point, Direction, length1: Length, length2: Length, orientation: Boolean): Line
Creates a line. Extrusion of a point depending on a direction.
- **revolve**(Point, axis, Line: angle1, Angle: angle2, Angle): Circle
Enables to create a circle by revolving a point according to a given direction.

Example:

Geometrical Set.1\Curve.3=

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

Part Measures

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

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

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

Example

Total_Volume=
smartVolume(PartBody)

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

Example

Total_Area=
smartWetarea(PartBody\Pad.1)

Plane Constructors

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

Example

Geometrical Set.1\Plane.1=

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

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

Example

Geometrical Set.1\Plane.1=

plane(1,0,0,50mm)

creates the plane of $X=50\text{mm}$ equation.

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

Example

Geometrical Set.1\Plane.1=

plane(Geometrical Set.1\Line.1,Geometrical Set.1\Line.2)

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

Example

Geometrical Set.1\Plane.1=

plane(Geometrical Set.1\Point.1,Geometrical Set.1\Line.1)

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

Example

Geometrical Set.1\Plane.1=

plane(Geometrical Set.1\Curve.1)

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

Example

Geometrical Set.1\Plane.1=

planetangent(Geometrical Set.1\Sweep.1, Geometrical Set.1\Point.1)

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

Example

Geometrical Set.1\Plane.1=

planenormal(Geometrical Set.1\Spline.1, Geometrical Set.1\Point.1)

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

Example

Geometrical Set.1\Plane.2=

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

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

Example

Geometrical Set.1\Plane.2=

planeoffset(Geometrical Set.1\Plane.1, Geometrical Set.1\Point.1)

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

Example

Geometrical Set.1\Plane.2=

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

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

Analysis operators

- **energy** (*Case: StaticSolution*)
Computes the global energy in a static case solution.
- **misesmax** (*Case: StaticSolution*)
Computes the maximum value of the nodal VonMises stress.
Example
misesmax.1 = misesmax("Finite Element Model\Static Case Solution.1")
- **dispmax** (*Case: StaticSolution*)
Computes the nodal maximum displacement.
Example
length.1 = dispmax("Finite Element Model\Static Case Solution.1")
- **frequency** (*Case: FrequencySolution*)
Computes a given frequency.
Example
Frequency.1 = Frequency("Finite Element Model\Frequency Case Solution.1")
- **frequencies** (*Case: FrequenciesSolution*)
Computes all the frequencies.
Example
FrequenciesList.1 = Frequencies("Finite Element Model\Frequencies Case Solution.1")
- **globalerror** (*Case: StaticSolution*)
Computes the global error percentage of a static case.
Example
percentage.1 = globalerror("Finite Element Model\Static Case Solution.1")
- **bucklingfactors** (*Case: BucklingSolution*)
Computes a list of buckling factors.
Example
Bucklingfactors.1 = BucklingFactors("Finite Element Model\Buckling Case Solution.1")
- **dispmaxongroup** (*Case: AnalysisResults, Group: Group*): Length
Computes the nodal maximum displacement. It applies to a group of items.

Mathematical Functions

Sample (illustrates interpolations): [KwrInterpolations.CATPart](#)

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

The returned number is 13 mm

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

- **CubicInterpolation**(*arg1*:Real, *arg2*:Real, *arg3*:Real) : Real
Should be used when creating a parallel curve from a law.
Example:
1 - Create a line in the Generative Shape Design workbench
2 - Access the Knowledge Advisor workbench and create the law below:
FormalReal.1 = CubicInterpolation(1,50,FormalReal.2)
3 - Back to the Generative Shape Design, create a parallel curve. Select the Law mode and specify the law above as the one to be applied.
- **mod(Real,Integer): Real**
Enables the user to retrieve the remainder of the division of the integer part of the real number by the integer.
- **Cos(Real):Real, cosh (Real): Real**
Calculates the cosine(cos) or hyperbolic cosine(cosh).
Example
Real.1 = cos(PI* 1rad/4)
Real.1 = cos(45deg)
- **tan(Real): Real, tanh(Real): Real**
Calculates the tangent(tan) or hyperbolic tangent (tanh).
- **sin(Real):Real, sinh(Real):Real**
Calculates the sine or hyperbolic sine.
- **asin(Real):Real, asinh(Real):Real**
Calculates the arcsine or hyperbolic arcsine.
- **acos(Real):Real, acosh(Real):Real**
Calculates the arccosine or hyperbolic arccosine.
- **atan(Real):Real, atanh(Real):Real**
Calculates the arctangent or hyperbolic arctangent.



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

Creating a Formula




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



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



1. Open the [KwrStartDocument.CATPart](#) document.

2. Click the  icon to display the [f\(x\) dialog box](#). Make sure that the Incremental box is unchecked.

Method 1

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

Method 2

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

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

 - Click **OK** in the Formulas Editor.
- 3.** Click **Apply** to update the document.
 - 4.** Click **OK** to close the dialog box.






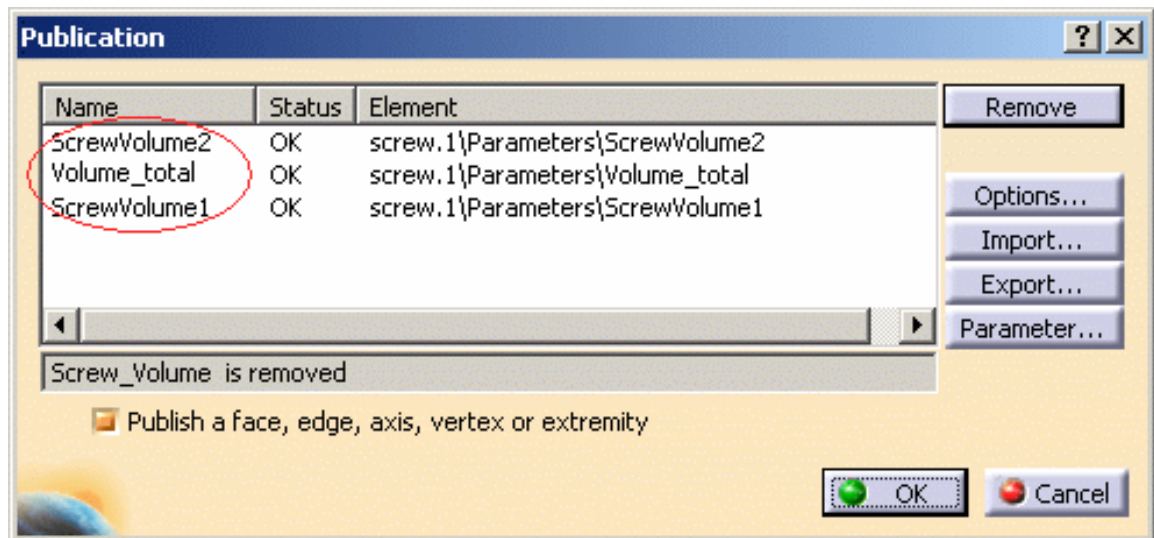
Creating Formulas based on Publications



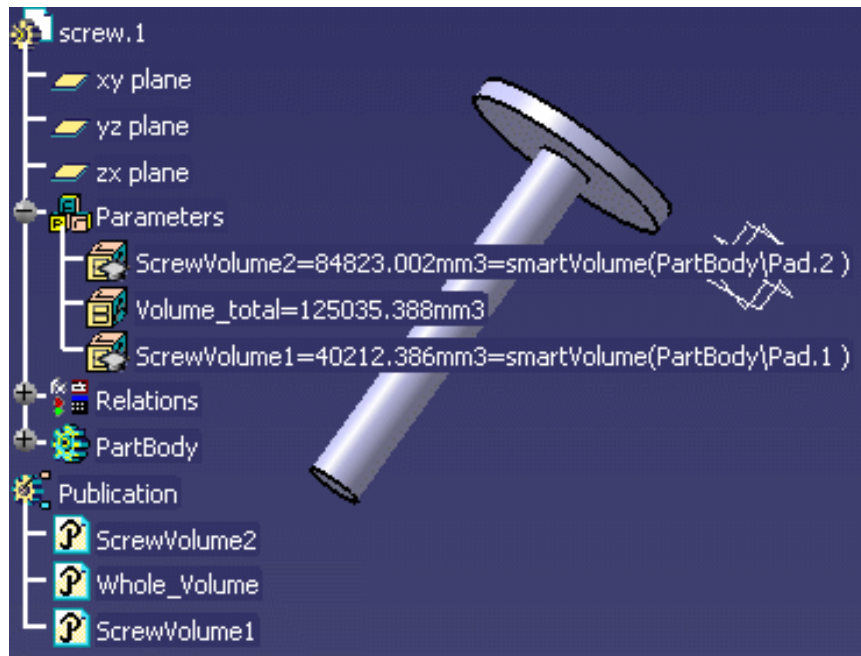
This task explains how to create a formula based on publications in a CATProduct file.



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

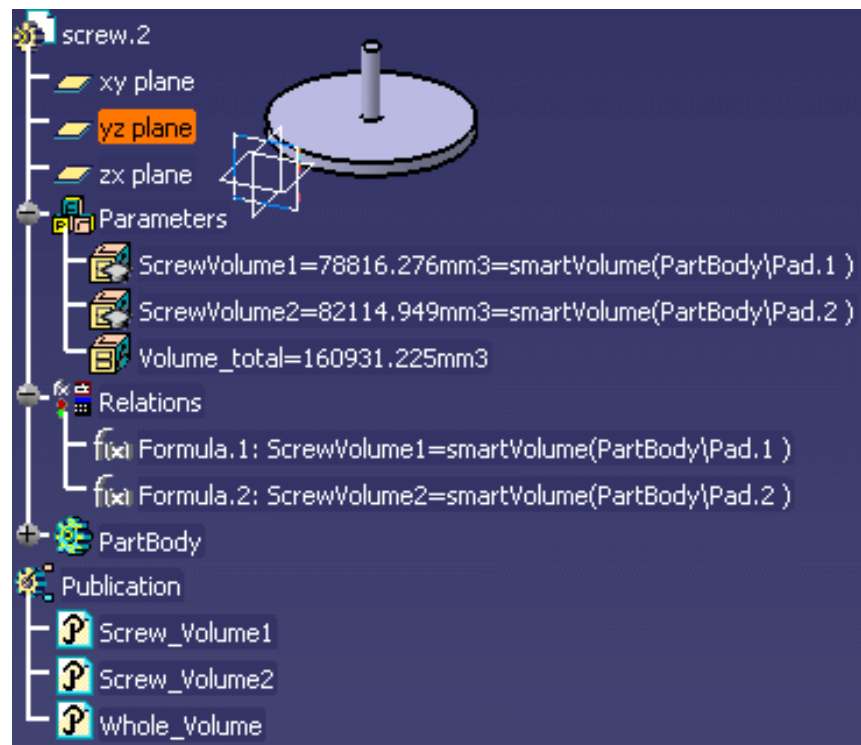


6. Click **OK** when done. The parameters, the formulas, and the publications are created (see graphic below).



7. Save your file and close it.

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




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

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

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

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

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

Specifying a Measure in a Formula



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

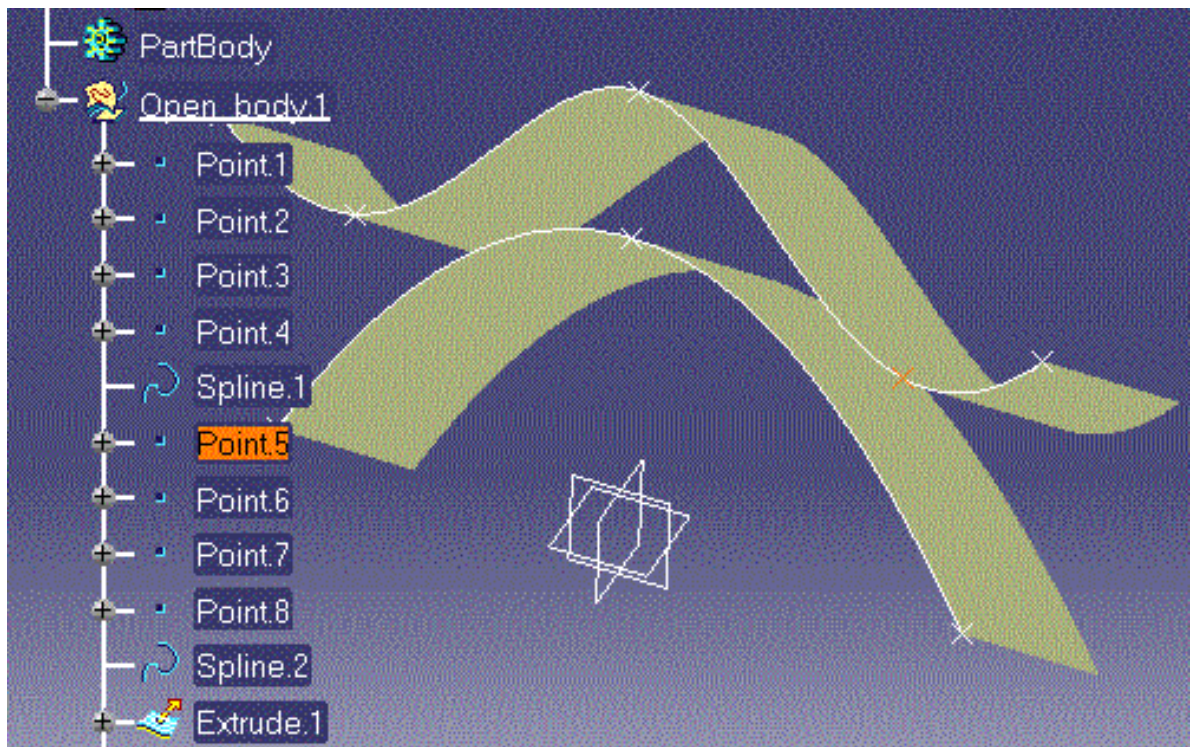


Measures, i.e. values captured from the geometry area can be used in formulas. Here are some examples of measures that can be used in formulas:

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



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

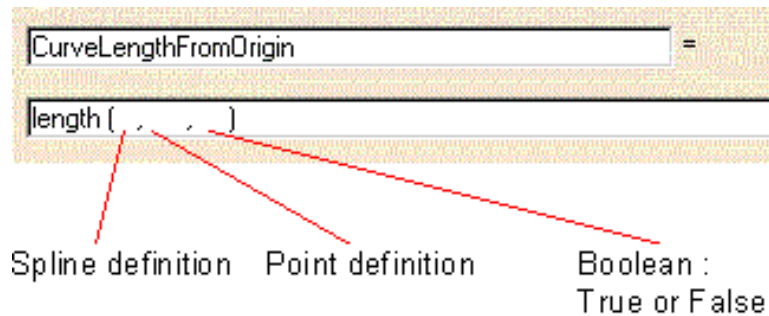


3. Click the Formula icon. The $f(x)$ dialog box is displayed.
4. Create the CurveLengthFromOrigin parameter. To do so, proceed as follows:

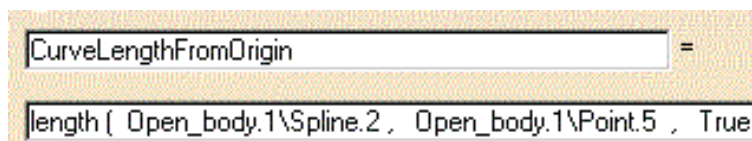
- o Select the Length item with Single Value in the New Parameter of type list, then click New Parameter of type. The new parameter appears in Edit name or value of the current parameter.
- o Replace the Length.1 name with CurveLengthFromOrigin, and click Apply.

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

- a.** Select the CurveLengthFromOrigin parameter in the parameters list, then click Add Formula. The Formula editor is displayed.
- b.** Select the Measures item from the Dictionary list.
- c.** In the list of measures, double-click the length(Curve,Point,Boolean) item. The length function is added to the Formula Editor.



- d.** Fill in the Formula editor field as indicated below.
 - 1.** The three arguments are: a curve to be selected from the geometry area, a point to be selected from the geometry area and a boolean.
 - 2.** Position the cursor where the first argument is intended to be typed. Then double-click the Spline.2 feature in the specification tree. The curve argument is added to the length definition.
 - 3.** Position the cursor where the second argument is intended to be typed. Then double-click the Point.5 feature in the specification tree. The point argument is added to the length definition.
 - 4.** Type a boolean for the third argument: True if the length is to be calculated from the origin, False if the length is to be calculated from the curve end.



- 5.** Click **OK** to confirm the formula definition. You are back to the Formulas dialog

box. The CurveLengthFromOrigin formula and value(47.5mm) are added to the parameter list.

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



Referring to External Parameters in a Formula



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




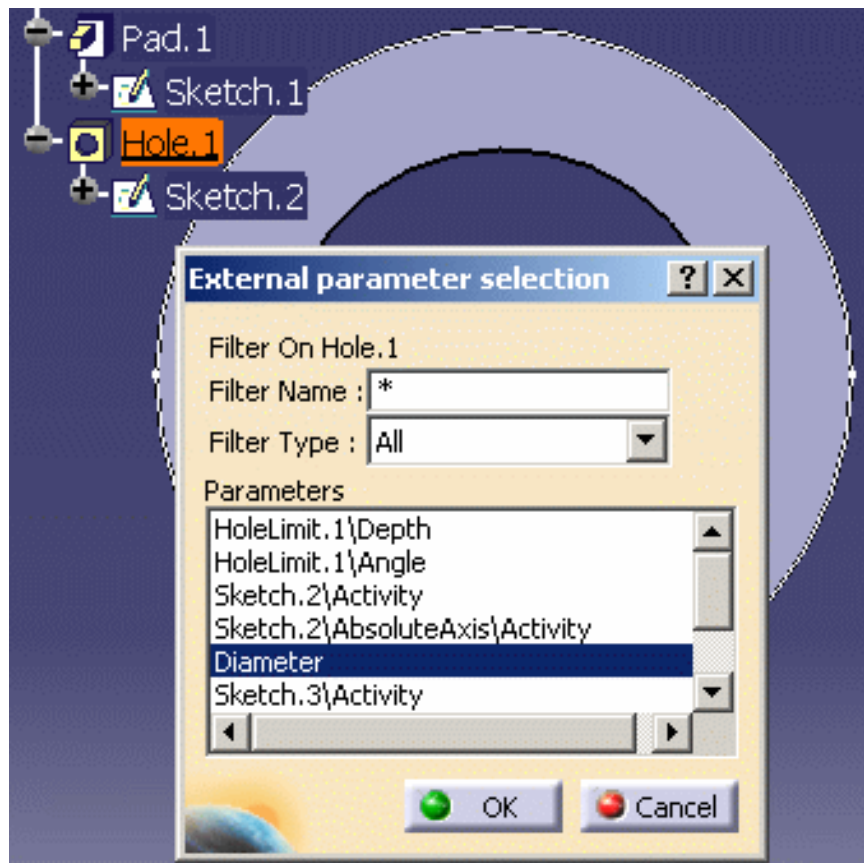
In a formula, you can use parameters defined in external documents. This works between any types of document. For example, in a CATPart document, you can specify a formula referring to parameters defined in a CATDrafting document. External parameters can also be used when working within an assembly.



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



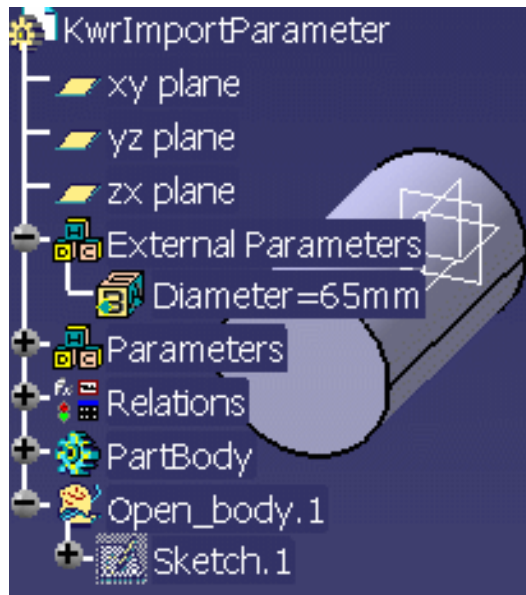
1. Open the [KwrStartDocument.CATPart](#) document as well as the [KwrImportParameter.CATPart](#) document. Select the **Window->Tile Vertically** command from the standard menu bar. Both documents are displayed.
2. Make active the KwrImportParameter document. Click the  icon to display the [f\(x\)](#)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



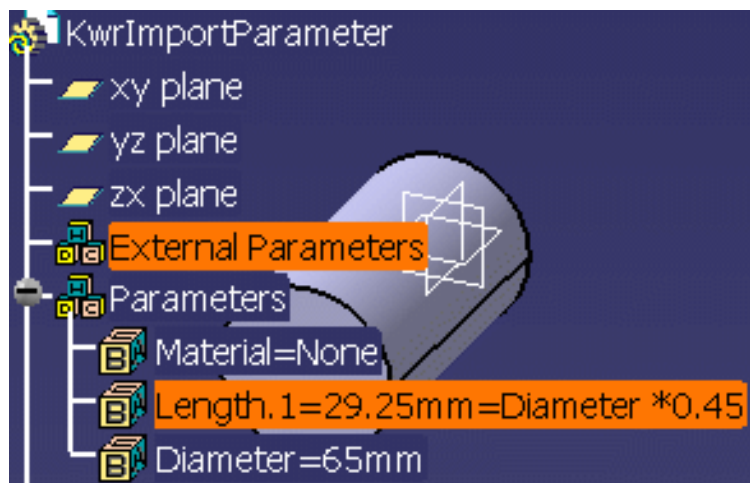
5. In the **External parameter selection** dialog box, select the Diameter object in the external parameter list. Then click OK. The Length.1 definition is carried forward to the formula editor. (Click the picture below to enlarge it.)



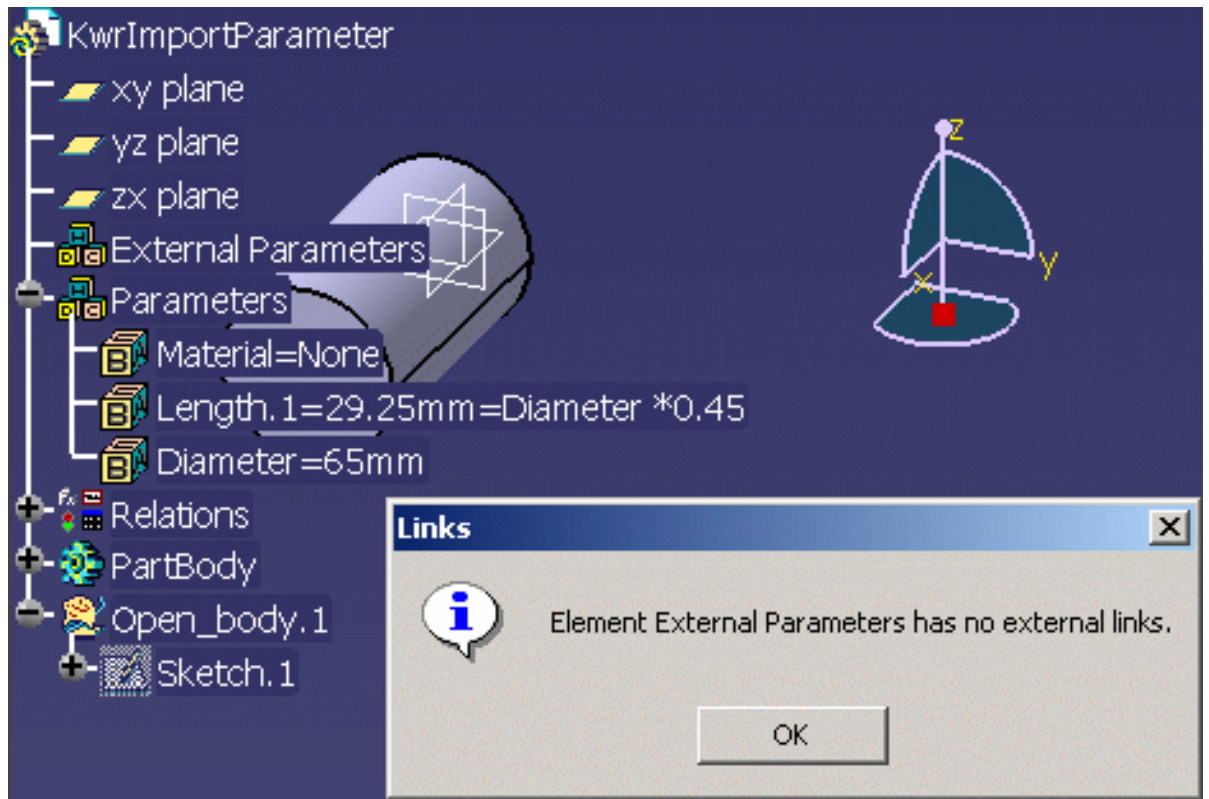
6. Complete the formula definition as indicated below:
$$\text{Length.1} = \text{Diameter} * 0.45$$
7. Click **OK** in the formula editor. You are back to the Formulas dialog box. In the parameter list, the Length.1 parameter value is modified according to the formula specified. In the KwrImportParameter specification tree, the External Parameters node is added. Expand this node to display the Diameter parameter.




8. Click **OK** to add the formula to the KwrImportParameter.CATPart document and exit the dialog.
9. Select the Edit->Links command from the standard menu bar. The displayed dialog box confirms that there is a link between the KwrImportParameter\Length.1 object and the KwrStartDocument\PartBody\Hole.1\Diameter object.
10. Click **Isolate** in the Links dialog box, then click **OK**. In the KwrImportParameter.CATPart specification tree, the External Parameters node can no longer be expanded and the Diameter parameter is added below the Parameters node.



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




Using the Equivalent Dimensions Feature

- 

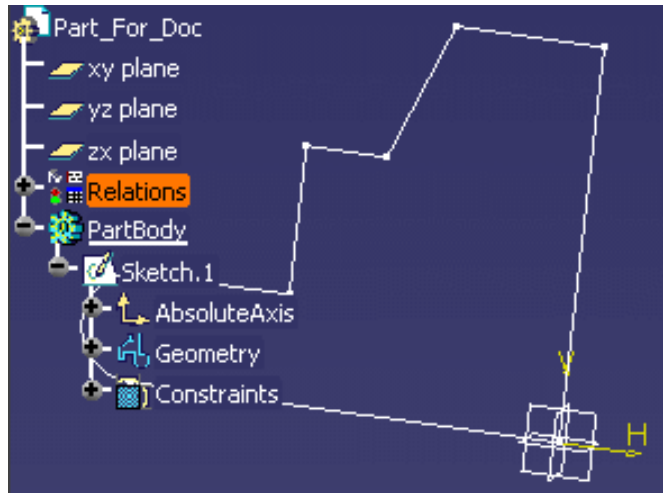
This scenario explains how to use the Equivalent Dimensions Feature. The scenario described below is divided into the following steps:

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


 To know more about the Equivalent Dimensions feature, see [Getting Familiar with the Equivalent Dimensions Interface](#).

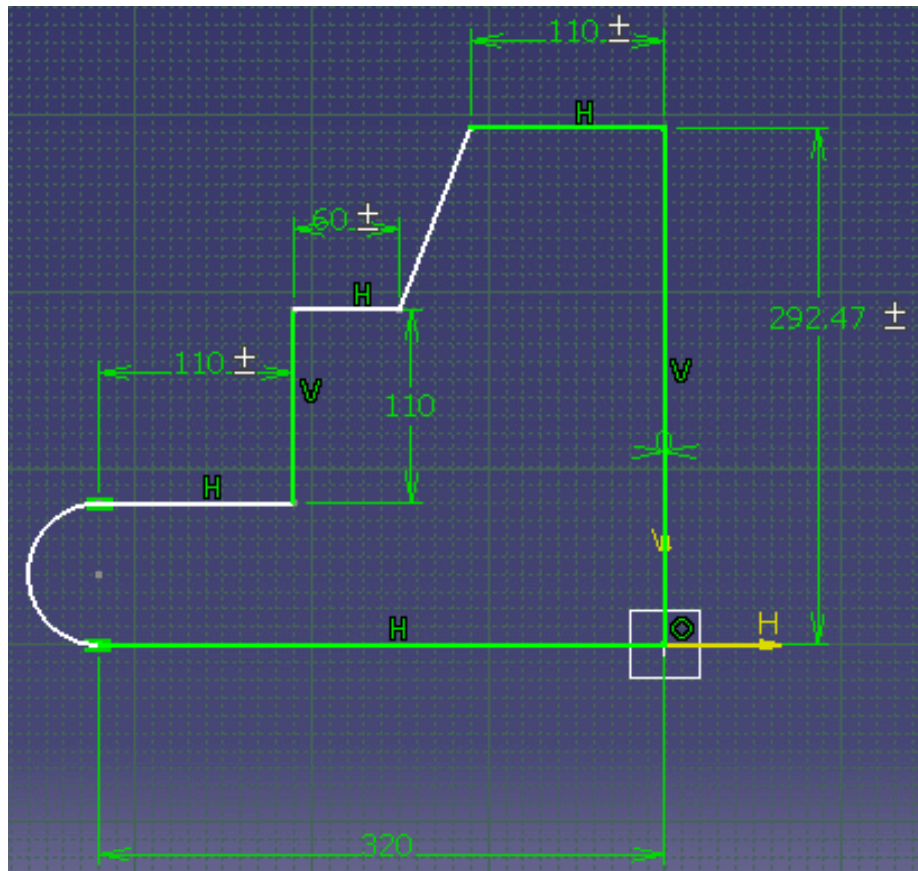
- 


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

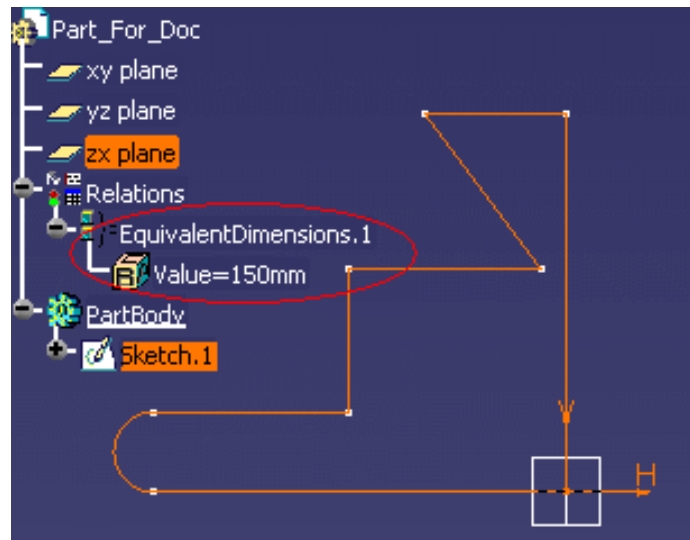


2. In the specification tree, expand the PartBody node and double-click Sketch.1 to access the sketcher.

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



4. In the Knowledge toolbar, click the **Equivalent Dimensions** icon (). The Equivalent Dimensions Edition window displays.
5. Click the **Edit List...** button. In the opening window, use the arrow key to select the following parameters and click **OK** when done.
 - o Length.34
 - o Length.36
 - o Length.37
6. In the Equivalent Dimensions Edition window, set the value to 150mm and click **OK**.
7. Exit the Sketcher. The sketch is modified accordingly and the EquivalentDimensions.1 feature displays below the Relations node.




8. Double-click Value= 150mm twice in the specification tree. The **Edit Parameter** window displays.
9. Enter 140mm and click **OK**.




Getting Familiar with the Equivalent Dimensions Interface

- [The Equivalent Dimensions Interface](#)
- [The Equivalent Dimensions Contextual Menu](#)

The Equivalent Dimensions Interface

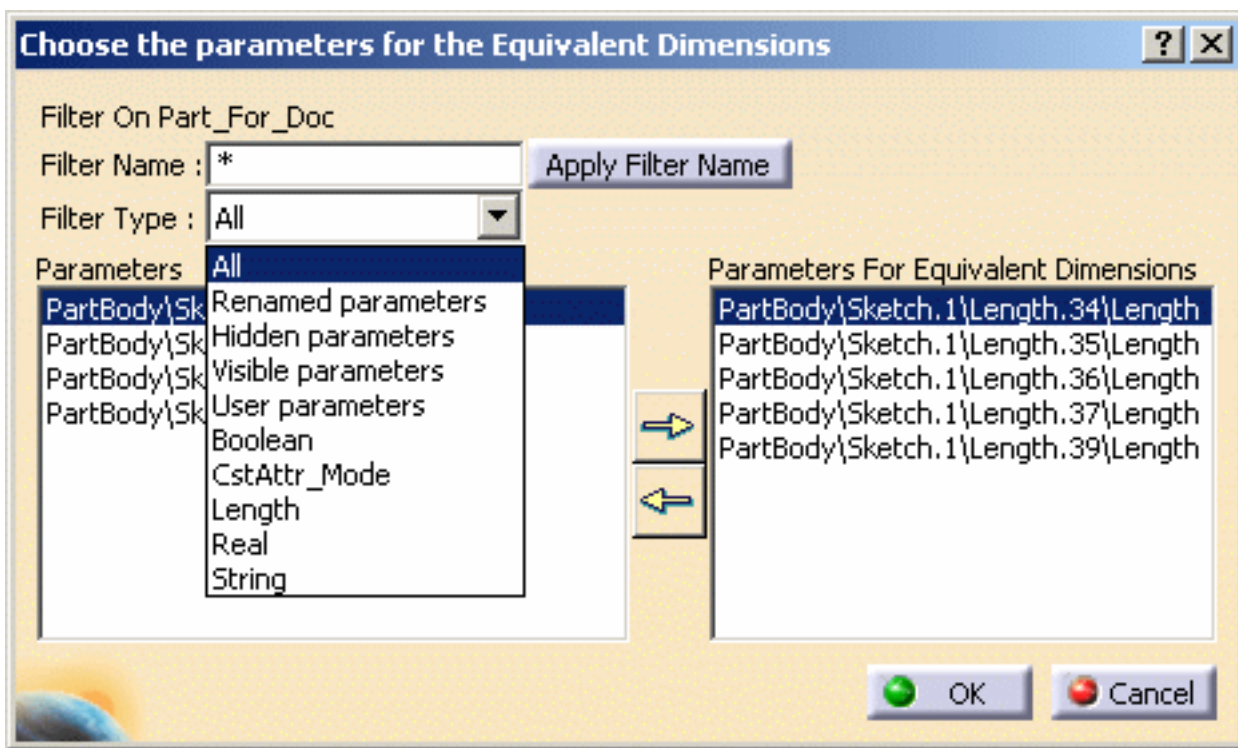
The user can create an Equivalence Dimensions feature. This new feature can be accessed by clicking the **Equivalent Dimensions** icon () in the Knowledge toolbar. It is the equivalent of a formula applied to Length parameters. It is designed to enable the user to apply the same value to selected parameters. It can be used with the following parameters:

- 3D parameters
- Sketch parameters

 Note that this new command cannot be used in association with parameters valuated by a formula.

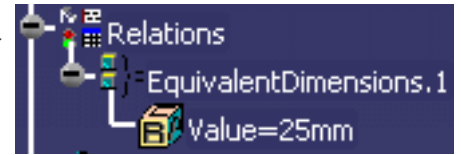
To create this feature, proceed as follows:

- click the **Equivalent Dimensions** icon () . The Equivalent Dimensions window displays.
- Click the **Edit List...** button. The following image displays:

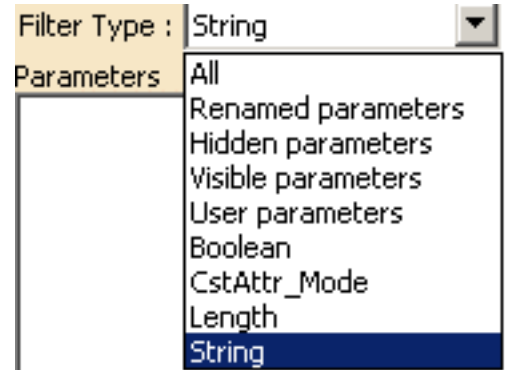


i The **Apply Filter Name** function enables the user to perform a search on a given string. The result of his search displays in the Parameters list. Note that if the user selects a 3D feature in the document, only the parameters of this feature will display in the Parameters list.

- In the Parameters list, use the arrow key to select the parameters that will have the same value.
- Click **OK** when done. The EquivalentDimensions feature displays below the Relations node as well as the value assigned to the selected parameters.

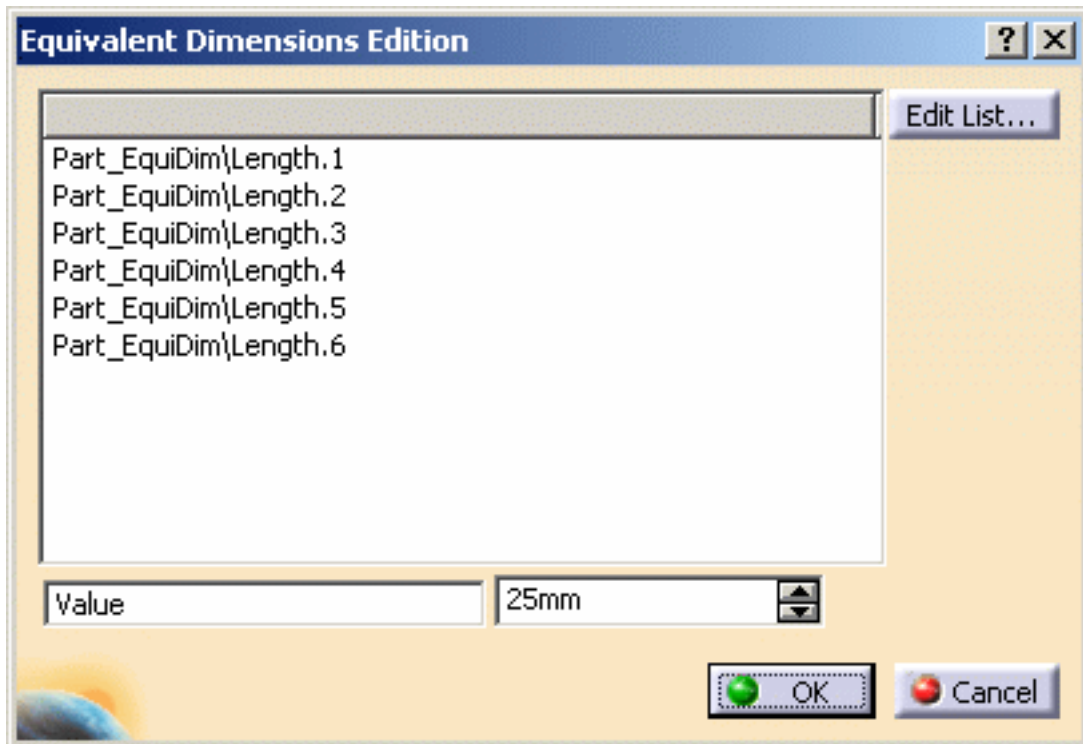


- i**
- Note that the **Filter Type** scrolling list enables the user to filter the parameters that display in the Parameters column. Select the **Length** filter when using the Equivalent Dimensions feature to display the Length parameters of a pad, or a sketch for example.



This feature can be edited by double-clicking it in the specification tree so that the user can:

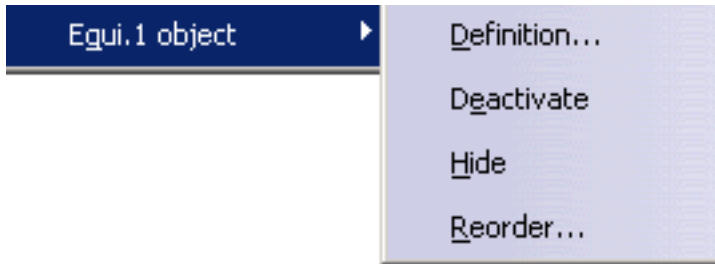
- Display the list of parameters belonging to the equivalent dimensions feature.
- Add or remove parameters from the equivalence list.
- Modify the value of the parameters.



Note that the value of the parameters belonging to the equivalence list changes when you change the value of one of the parameters of the list.

The Equivalent Dimensions Contextual Menu

You can access the Equivalent Dimensions contextual menu by right-clicking the equivalent dimensions feature in the specification tree.



The **Definition...** command enables you to access the Equivalent Dimensions Edition window.

The **Deactivate...** command enables you to deactivate the equivalent dimensions feature. In this case an icon indicates that the feature is disabled. To enable it, right-click it and select the **Activate...** command.

The **Hide** command enables you to hide the equivalent dimensions feature. In this case, it will not display in the specification tree.

The **Reorder...** command enables you to reorder the equivalent dimensions features (if you have created more than one of these features.)

Formulas: Useful Tips

The Incremental option of the formula editor

The Incremental option allows you to restrict the list of parameters displayed in the dictionary. Select a feature either in the tree or in the geometry area. Only the first level of objects right below the selected feature will be displayed in the dictionary. If the Incremental option is unchecked, all the objects below the selected feature are displayed.

The Incremental mode is useful when you work with large documents and when the parameter lists are long.

Tips about the formula editor

To help you write a formula, the formula editor provides you with a dictionary. This dictionary exposes the list of parameters and functions you can use to define a formula. Depending on the category of objects to be referred to in the formula, the dictionary is divided into two or three parts. To insert any definition in the formula editor, just double-click the object either in the dictionary or in the tree. If you double-click a function in the dictionary, its signature is carried forward to the formula editor. Only the argument definitions are missing.

Design Tables

Introducing Design Tables

Getting Familiar with the Design Table Dialog Box

Creating a Design Table from Current Values

Creating a Design Table from a Pre-Existing File

Interactively Adding a Row To the Design Table
External File

Regenerating a File From a Design Table

Controlling Design Tables Synchronization

Storing a Design Table in a PowerCopy

Optimal CATIA PLM Usability for Design Tables

Design Tables: Useful Tips



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

Introducing Design Tables

A design table:

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

Example

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

The Excel Sheet Format (under Windows)

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



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

If no unit is mentioned within a cell:

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

Here is an example of an Excel sheet:

column name

column unit

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

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

Within a given column, you can change the units.

Units can be specified in cells.
No unit = SI

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

The Tabulated Text File Format

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

```

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

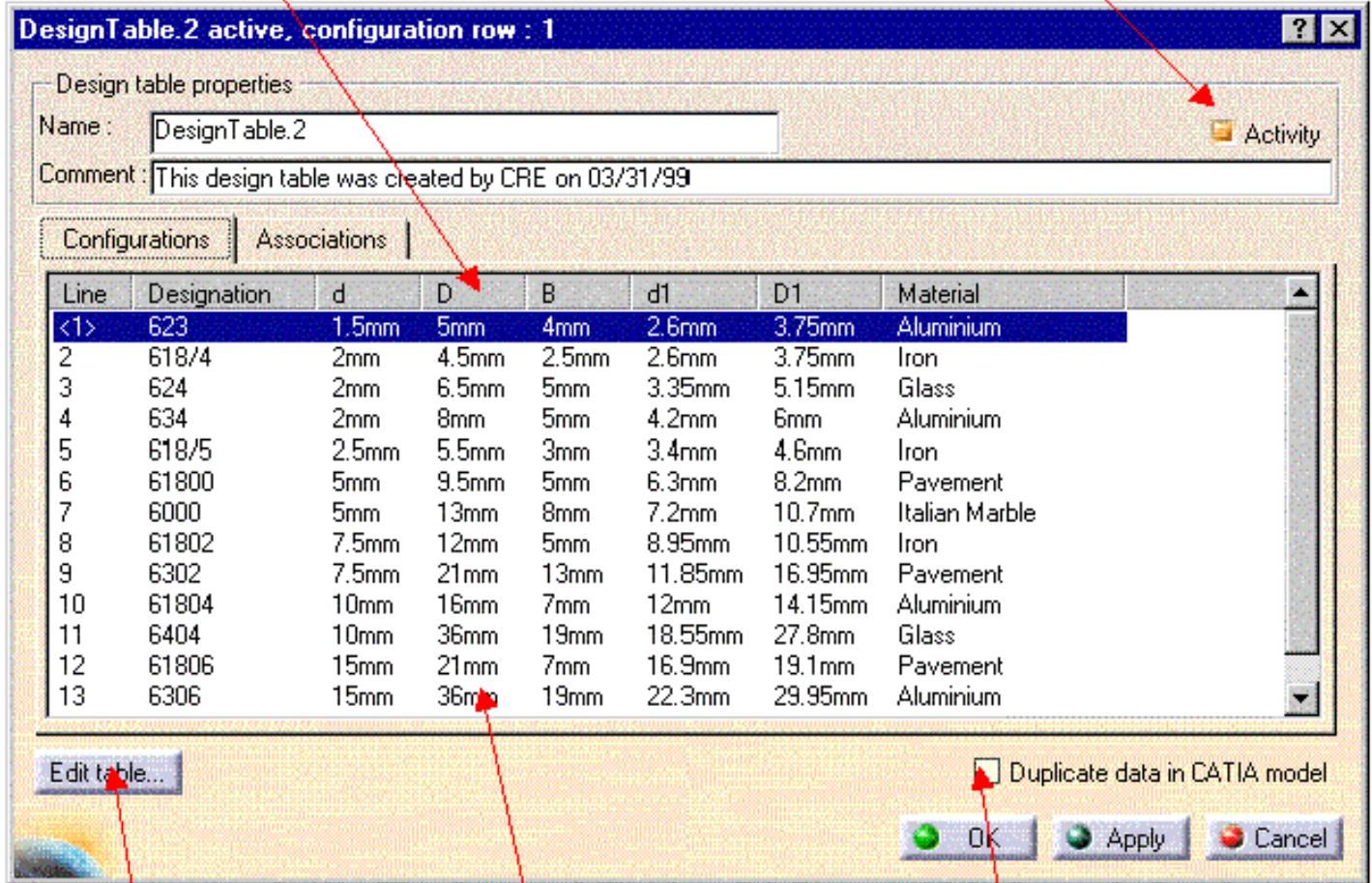
Under UNIX, it is possible to change the default design table editor. To do so, type:
export CATTextEditorDT=... (indicate the path of the editor.)

The CATIA Design Table

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

No units in column

Check box to modify the activity




Displays the design table raw data.

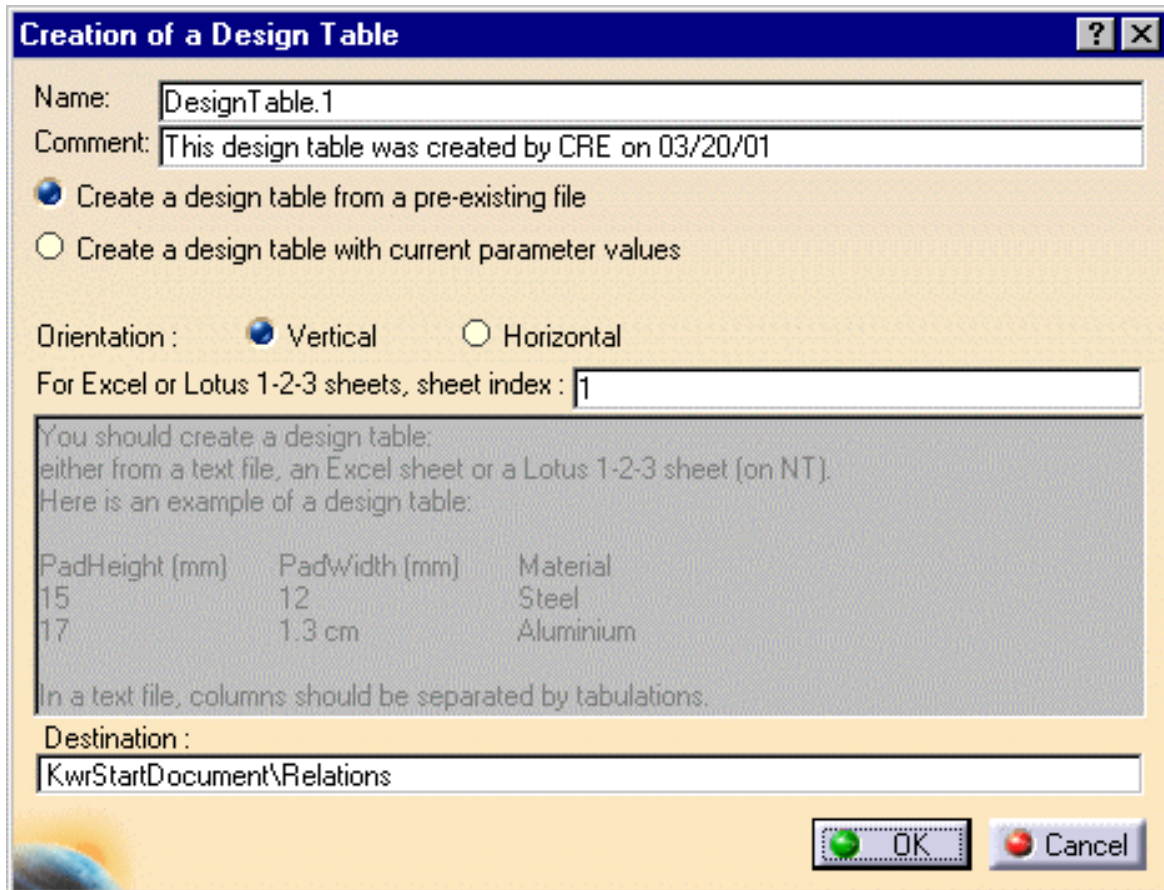
Values with units

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

Getting Familiar with the Design Table Dialog Box

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

Creation of a design table



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

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

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

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

Check this option whenever you want to create a design table from a subset of the document parameters. You just

have to select among all the document parameters the ones you want to be included as columns in the design table. In this case, the created design table only contains a single row.

The **Orientation** check boxes

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

The **sheet** index

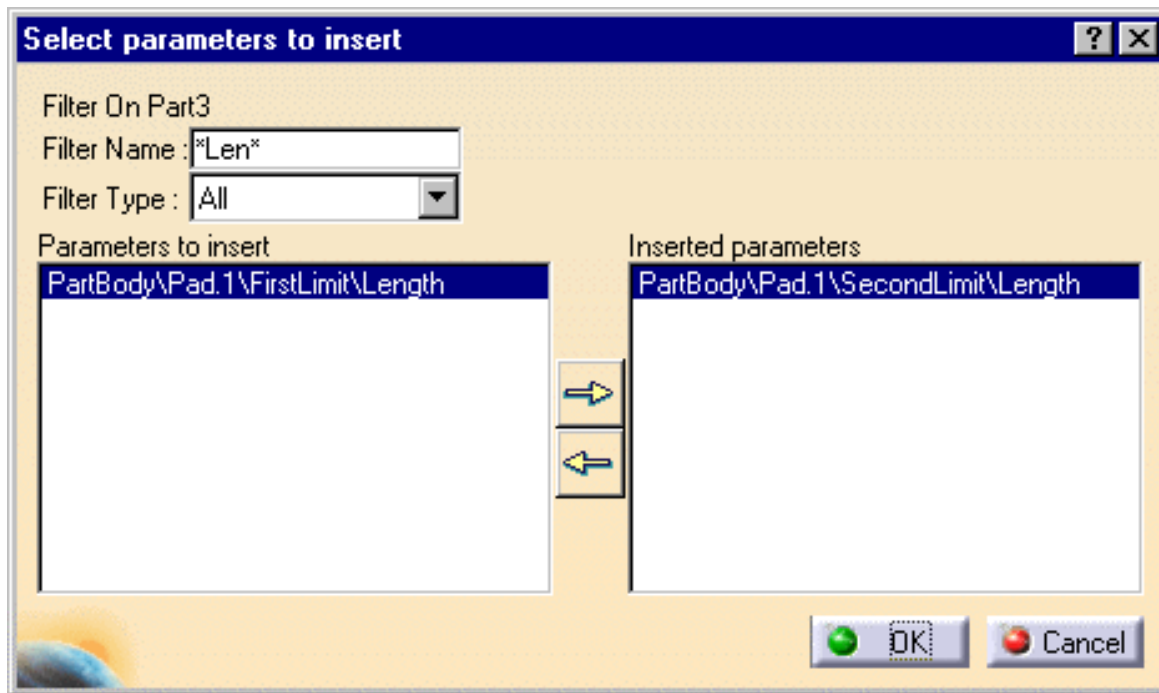
From Version 5 Release 7, you can specify an Excel or Lotus sheet number.

The **Destination** field

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

Selection of the parameters to insert

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

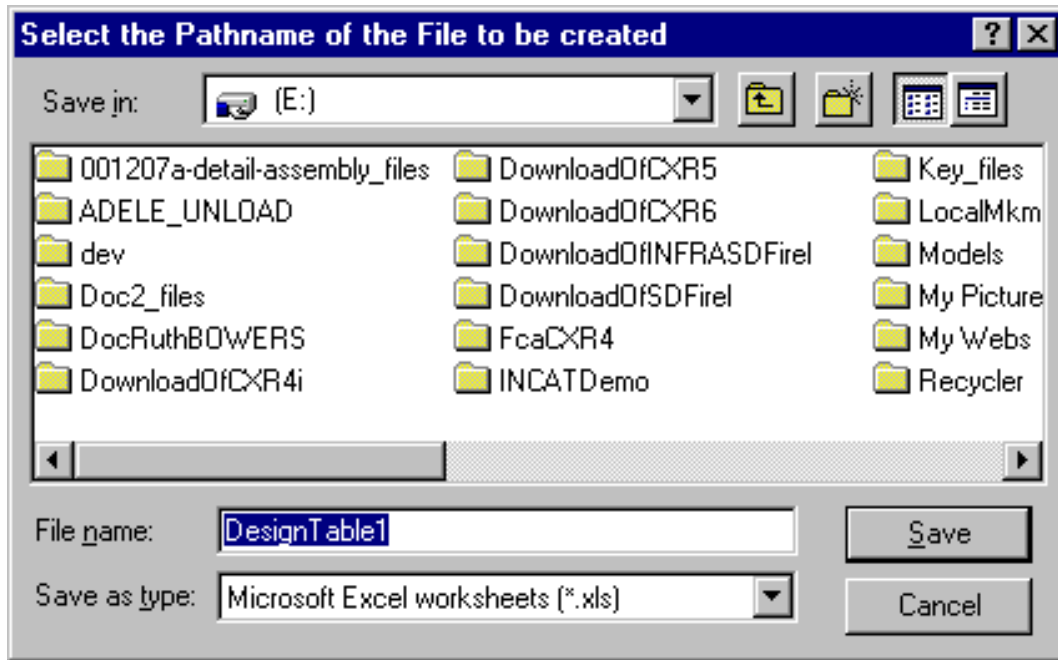


There are two ways to restrict the list of parameters to be displayed in the 'Parameters to insert' list. You can use the:

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

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

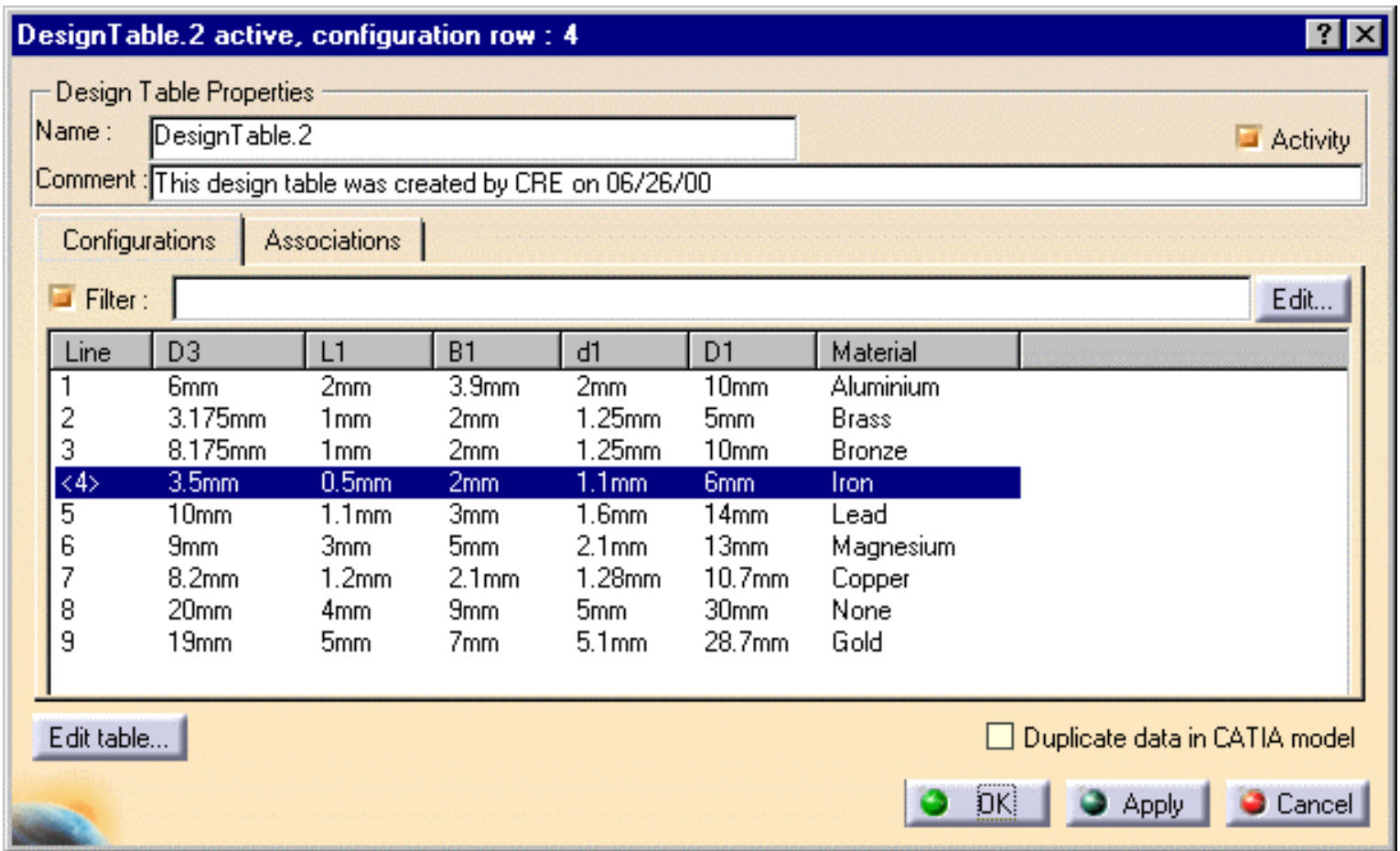
Selection of the file to be created



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

Design table dialog box

The 'Configurations' tab



The current configuration as well as its number (< configuration number >) are highlighted. To change the current configuration, you just have to click the new configuration in the design table.

A single row design table is created when you generate a design table with the current parameter values.

- **The Filter**

The filter is a means to help you query for a configuration meeting specific criteria. Click the "Edit..." button to display the "Design Table Request Editor". See [Using the Dictionary](#) for information on how to use the syntax provided by the dictionary.

In a query, you can specify a condition referring to the design table parameters as well as the parameters external to the design table.

- **The "Activity" check box**

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

- **The "Edit table..." push button**

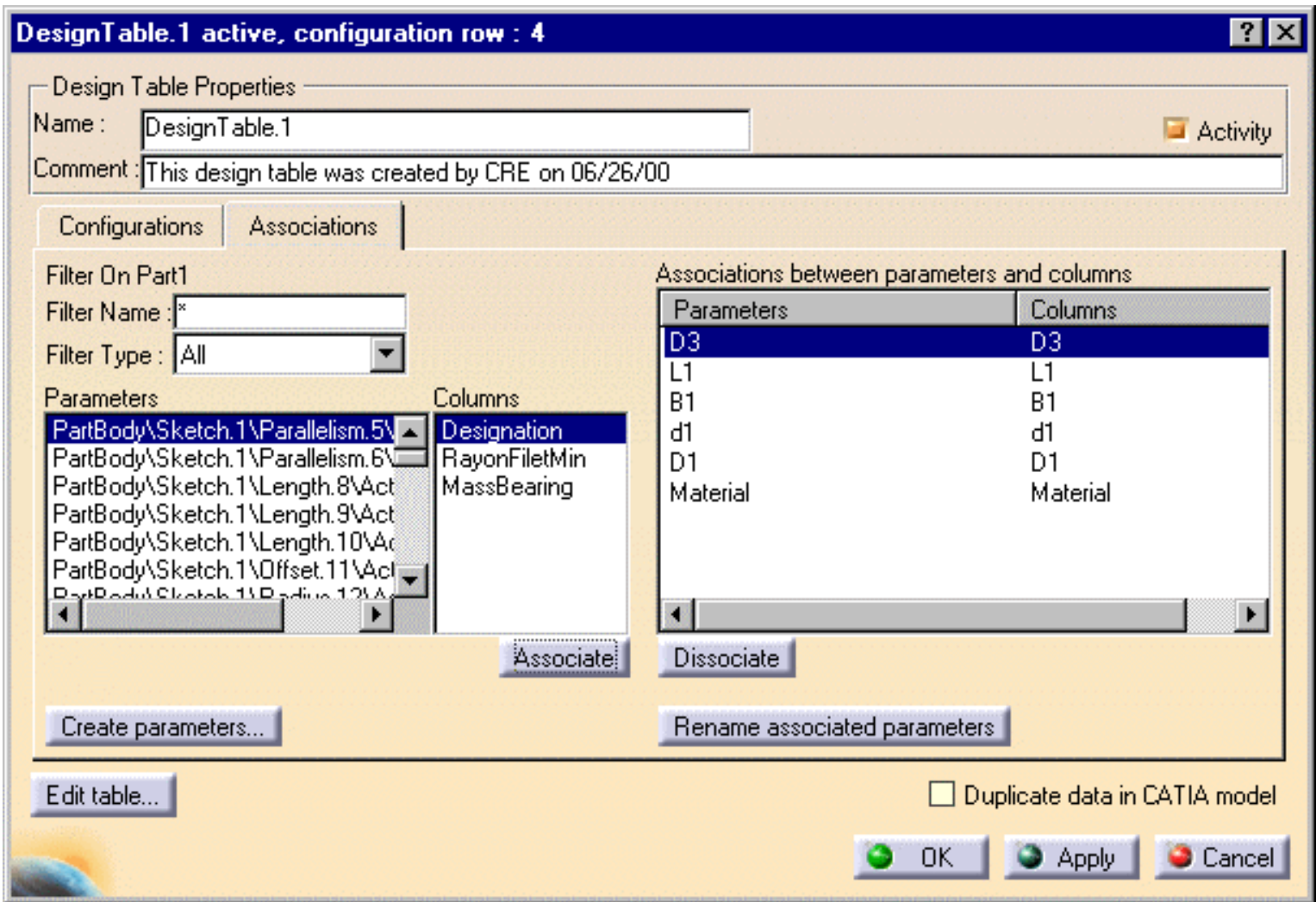
Click this button to display the edit table to be created. Depending on whether you have selected a .xls extension or not, you will launch a Microsoft Excel application or your default text editor for a .txt file.

- **The "Duplicate data in CATIA model" check box**

Check this box whenever you intend to reuse your document on an operating system different from the one used to create the design table. That way, your design table data will be duplicated into your document.

The "Associations" tab

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



- **The "Create parameters..." push button**

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

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

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

Creating a Design Table from the Current Parameters Values



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


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

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

- a. Create a table from the document parameters.
- b. Select the parameters to add to the design table.
- c. Specify a file to contain the generated design table.
- d. Edit the generated *CATIA* design table.
- e. Apply the design table to your document.

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



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

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

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



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

9. Click **Edit table...** to start an Excel application (under Windows) or open the text editor under Unix.
Replace the `PartBody\Pad.1\FirstLimit\Length` parameter value with 80mm.
10. Save your Excel or .txt file and close your application. Some information messages are displayed in a dialog box warning you about events related to the design table. Click Close.
11. Click **Apply** into the *CATIA* design table dialog, the document is updated as well as the *CATIA* design table. Click OK to exit the dialog and add the design table to the document.



Creating a Design Table from a Pre-existing File



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


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

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

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

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



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


The Iron parameter value is displayed in the specification tree.

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



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



1. Open the [KwrStartDocument.CATPart](#) document.
2. Click the  Design Table icon in the standard toolbar. The "Creation of a Design Table" dialog box is displayed. Enter a name (DesignTable2 for example) and a comment.
3. Check the Create a design table from a pre-existing file option. Click **OK**. A file selection panel is displayed.
4. Select the [KwrBallBearing.xls](#) file. Click **Open**. The Automatic associations dialog box is displayed.
5. Click **No**. The design table dialog box informs you that there is no associations between parameters and columns.

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

6. Click the **Associations** option. The table design dialog box now displays side by side the document parameter list and the input file columns.
7. In the Parameters list, select the PartBody\Hole.1\Diameter item. In the Columns list, select the d1 parameter. Then click **Associate**. A parameter couple is now displayed in the Associations between parameters and columns list.
8. Repeat the same operation for the Material parameter.

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



The parameter list can be filtered:

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

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

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

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



Interactively Adding a Row To a Design Table External File



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

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



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

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

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



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

[KwrAddARow.CATPart](#)

[KwrAddARow.xls](#)



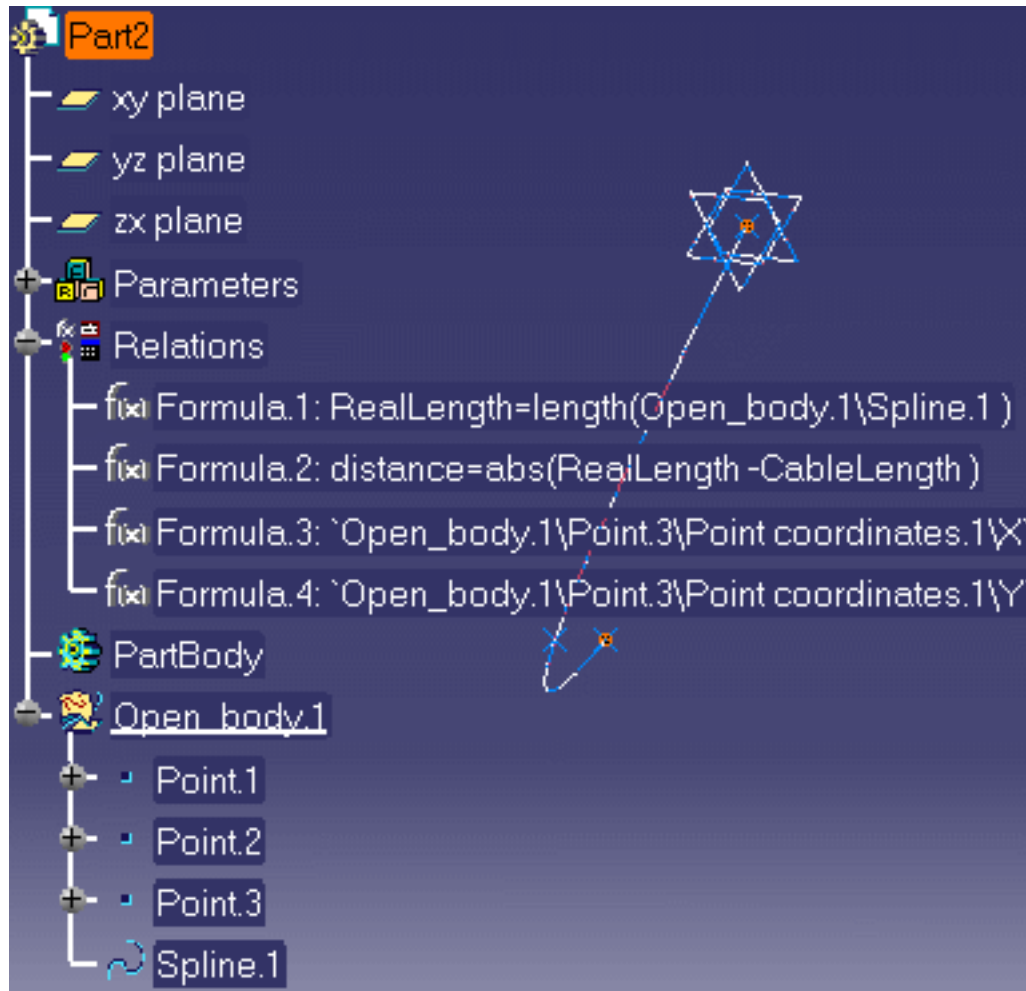
Note that this task can only be performed in an english environment.



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



1. Open the [KwrAddARow.CATPart](#) file. The following image displays.



2. Click the Design Table icon ()

3. Click the **Create a design table from a pre-existing file** radio button and click **OK**.

4. In the opening **File Selection** window, select the [KwrAddARow.xls](#) file and click **Open**.

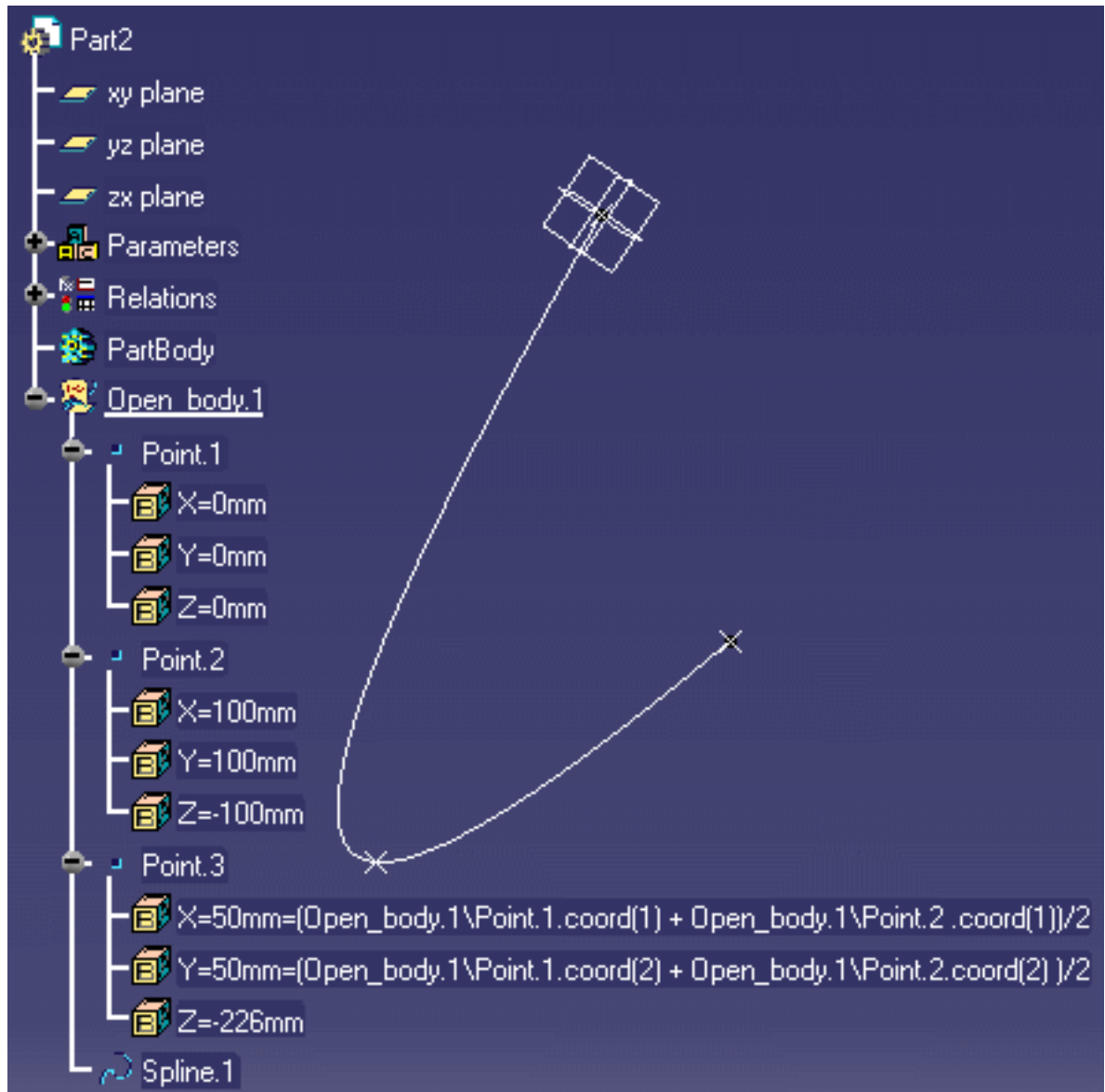
5. Click **Yes** in the Automatic associations window: The design table opens. Click **OK** to close it.

6. Click the **Measure update** icon to update Formula.1.

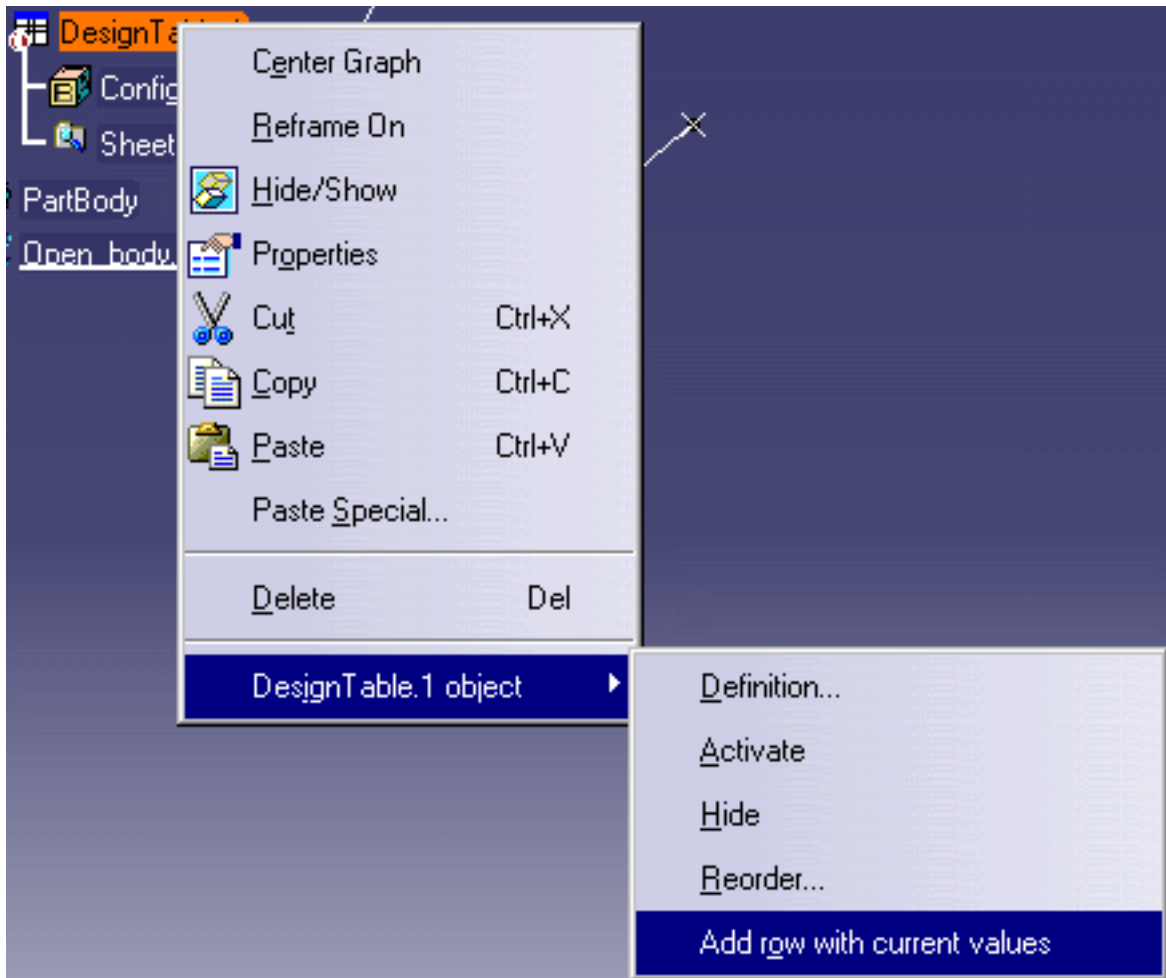
7. Under the Design Tables node, double-click **Configuration= 1**. The **Edit Parameter** dialog box displays.


8. Click the Design table icon in the **Edit Parameter** dialog box: The Design Table window displays.
9. In the dialog box, select the second configuration (line 2), click **Apply**, and **OK** twice.
10. Right-click Formula.1 in the specification tree and select the **Local Update** command.
11. In the Specification tree, right-click DesignTable.1 and select the **DesignTable.1 object->Deactivate** command. The design table is deactivated.
12. Modify the spline, to do so, proceed as follows:

<ul style="list-style-type: none"> Double-click Point.1 twice in the specification tree or in the Geometry. Enter the coordinates indicated below into the Point Definition dialog box. 				
<ul style="list-style-type: none"> Modify the coordinates of Point.2 and Point.3 (see table opposite) 		Point 1	Point 2	Point 3
	X	0	100	50
	Y	0	100	50
	Z	0	-100	-226
<ul style="list-style-type: none"> Click OK when done. 				



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



14. Right-click DesignTable.1 and select the **DesignTable.1 object**->**Activate** command.
15. Double-click **Configuration= 1** under DesignTable.1 and click the Design Table icon () .
16. In the DesignTable.1 window select the configuration that you have just added and click **Apply** and **OK** twice. The spline is updated accordingly.



Generating a File From a Design Table



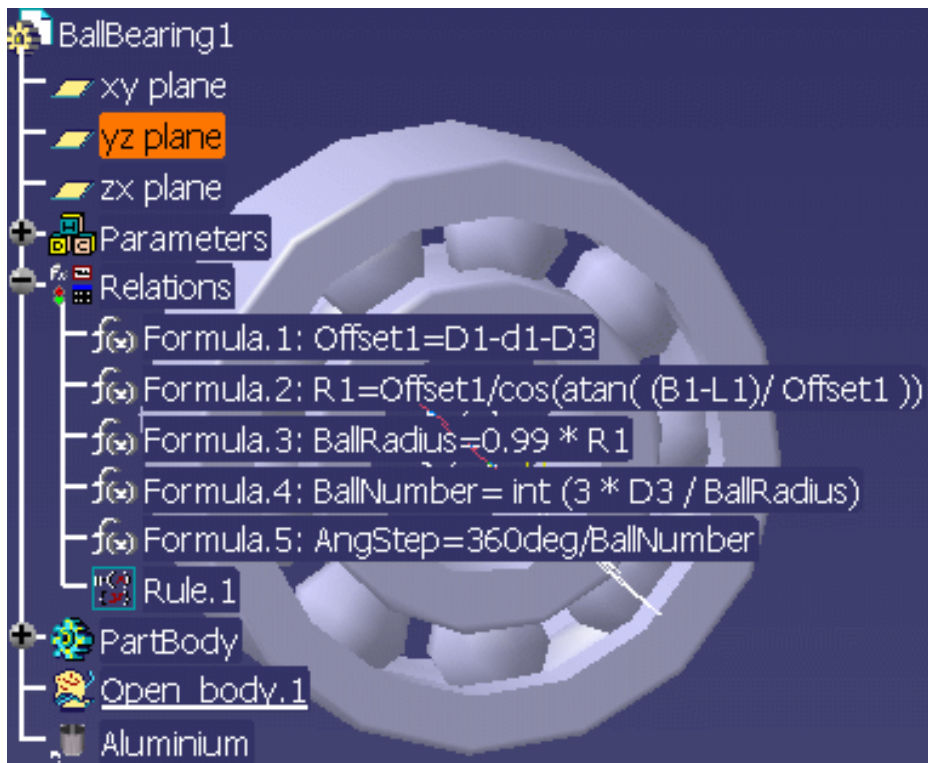
This topic explains how to regenerate an external file (.XLS or .txt format) using the data contained in the model. The data contained in the model come from an external file that was previously deleted. There are 2 ways to regenerate the file:


- **Using the Create New File command:** This command is available in the **Manage Design Tables** window which displays only if the **Interactive Synchronization at Load** option is checked in the Knowledge tab (**Tools->Options...->Parameters and Measure**).
- **Using the Export Content to file command**



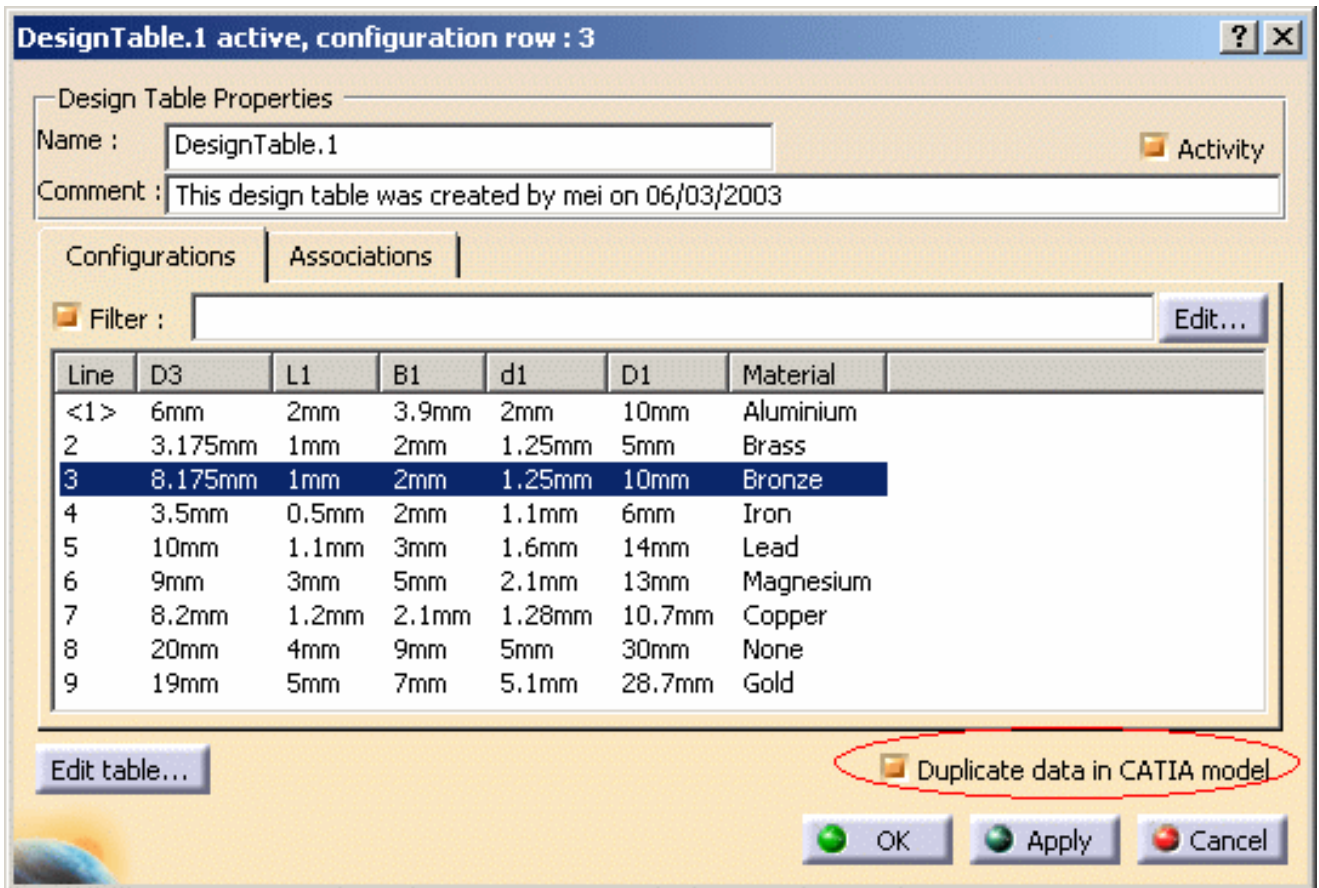
Using the Create New File Command

1. From the **Tools->Options...->Parameters and Measure** command, access the **Knowledge** tab and make sure the **Interactive Synchronization at Load** option is checked.
2. Open the **KwrBallBearing1.CATPart** file. The following image displays.

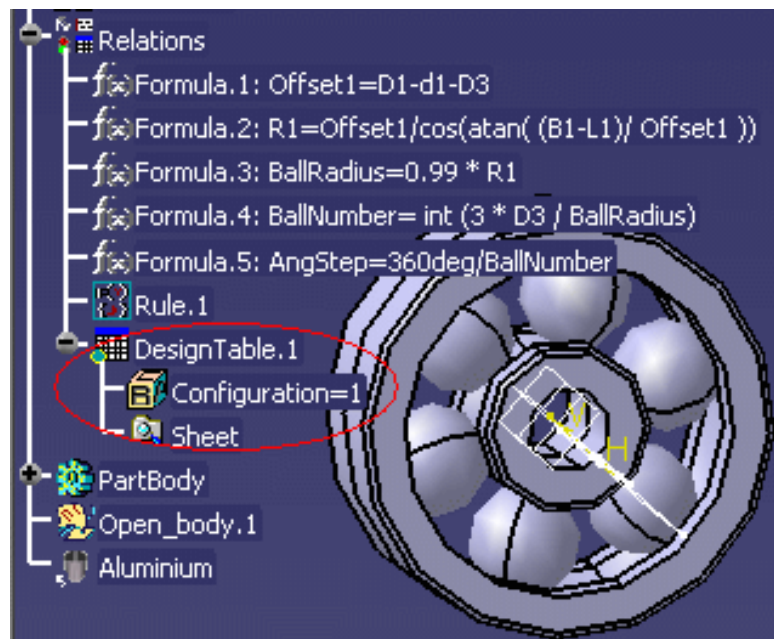


3. Click the **Design Table** icon () in the Knowledge tool bar. The **Creation of a Design Table** window displays.
4. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.

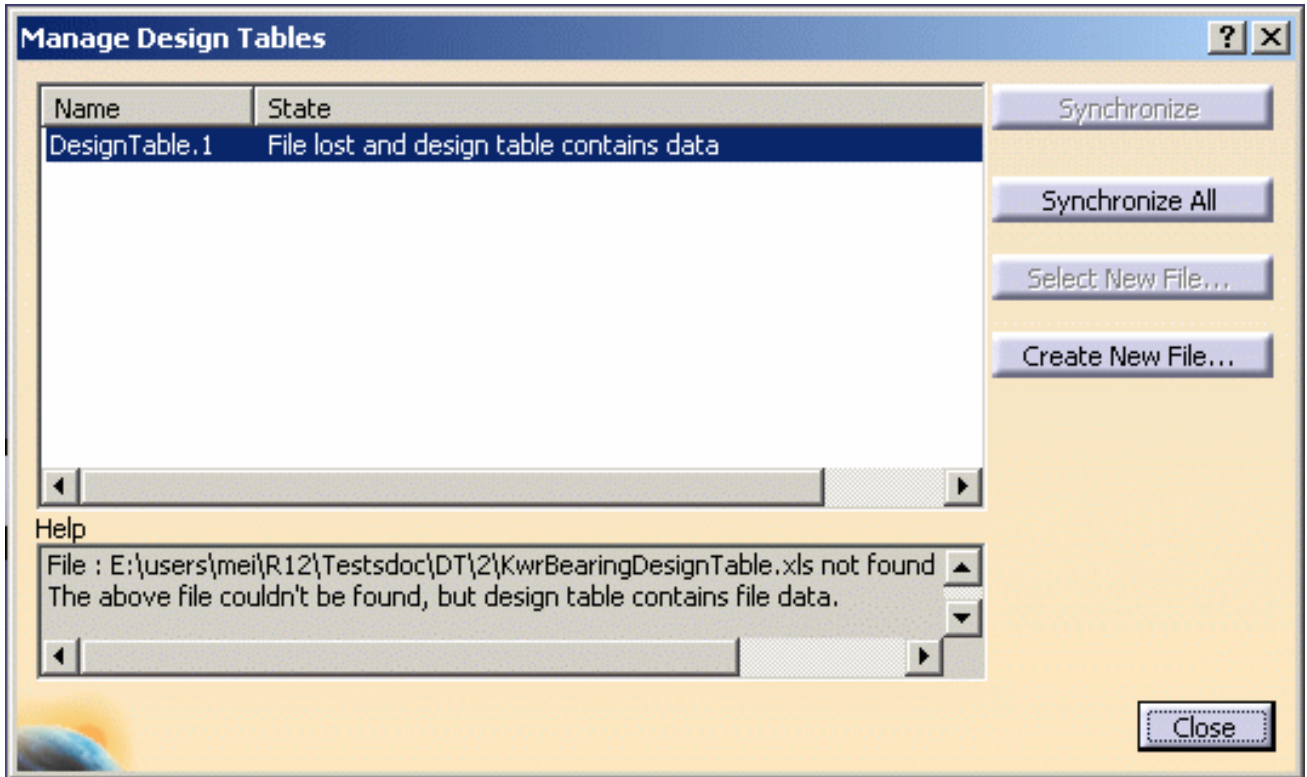
5. Select the [KwrBearingDesignTable.xls](#) file and click **Open**. Click **Yes** when asked if you want to associate the columns of the table with the parameters.
6. In the DesignTable.1 window, Check the **Duplicate data in CATIA model** option. Note that if this option is not checked, you will not be able to generate a new file.



7. Click **OK** to apply the default configuration. DesignTable.1 displays below the Relations node.



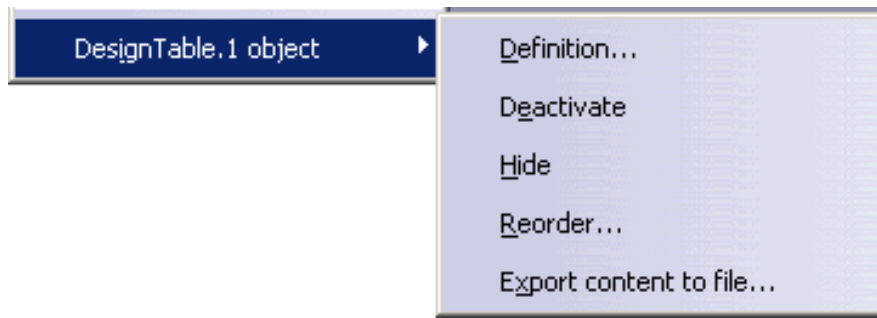
8. Save your file and close it.
9. Delete the KwrBearingDesignTable.xls file from its current location.
10. Open the KwrBallBearing1.CATPart file that you have just saved. The **Manage Design Tables** window displays indicating that the external file has been deleted.



11. Click the **Create New File ...** button to generate a file from the data contained in the .CATPart document. The Save As dialog box displays.
12. Enter the name of the file that you want to create: DT2 in this scenario. .XLS is the default file type. The text format is also available in the **Save as type:** list. Keep the default settings.
13. Click **Save** and **Close** when done. The DT2.xls containing the design table data is created.

Using the Export content to file... command

14. From the **Tools->Options...->Parameters and Measure** command, access the **Knowledge** tab and make sure the **Manual Synchronization** option is checked.
15. Repeat steps 2 to 9.
16. Right-click DesignTable.1 in the specification tree and select the **DesignTable.1 object->Export content to file...** command.



- 17.** The Save As dialog box displays.
- 18.** Enter the name of the file that you want to create: DT3 in this scenario. .XLS is the default file type. The text format is also available in the **Save as type:** list. Keep the default settings.
- 19.** Click **Yes** when asked if you want this file to become the new design table source file. The DT3.xls containing the design table data is created.



Controlling Design Tables Synchronization



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

- [Automatic Synchronization At Load](#)
- [Interactive Synchronization At Load](#)
- [Manual Synchronization](#)

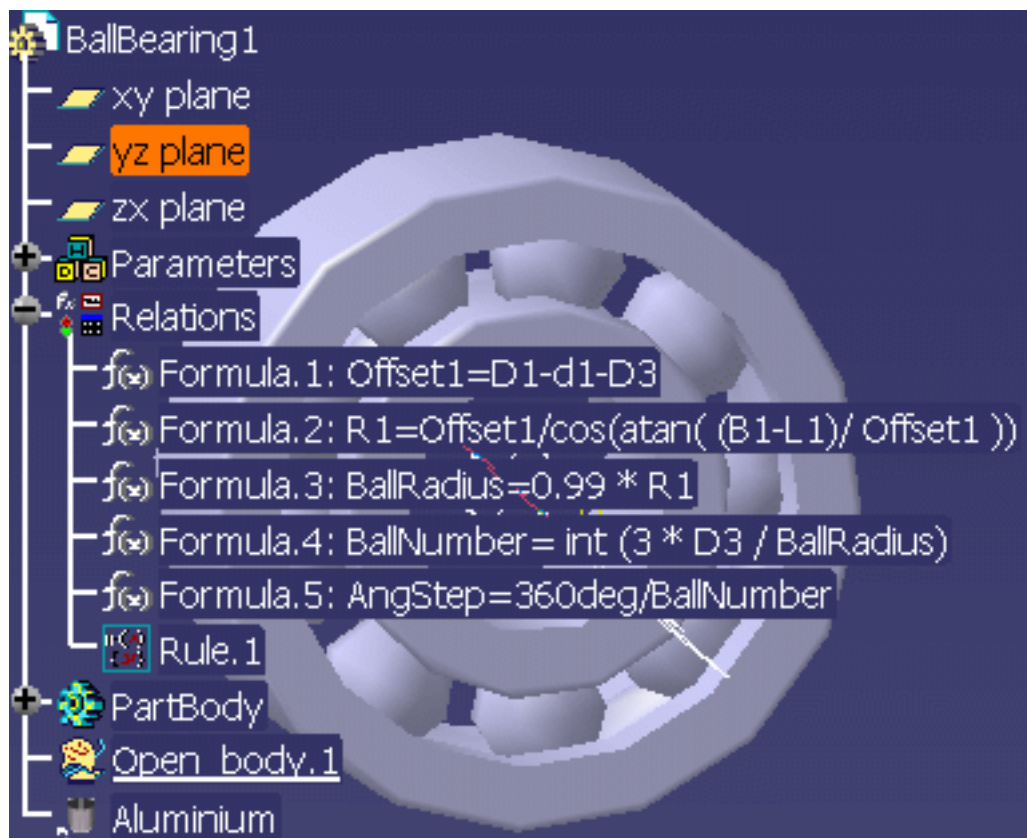



Automatic Synchronization At Load



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

1. Open the [KwrBallBearing1.CATPart](#) file. The following image displays.



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

3. Click the **Create a design table from a pre-existing file** option and click **OK**.
The File Selection dialog box opens.
4. Select the [KwrBearingDesignTable.xls](#) file and click **Open**. Click **Yes** when asked if you want to associate the columns of the tables with the parameters.
5. Click **OK** to apply the default configuration.
6. Save your file and close it.
7. Open the [KwrBearingDesignTable.xls](#) file. Change the material of row 2 to Gold. Save your file and close it.
8. Go back to Catia. Open the part: The Part is updated accordingly to your changes.

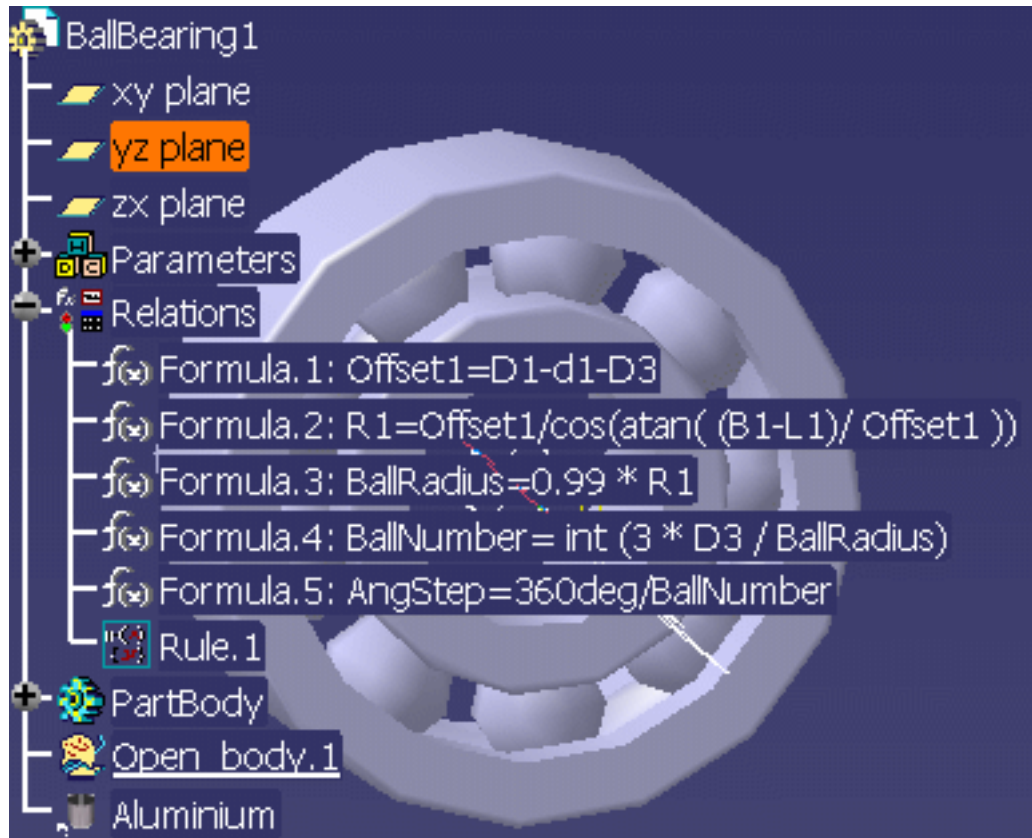



Interactive Synchronization At Load



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

1. From the **Tools->Options...** menu, select **General->Parameters and Measure** and check the **Interactive Synchronization At Load** option in the Knowledge tab.
2. Open the [KwrBallBearing1.CATPart](#) file. The following image displays.



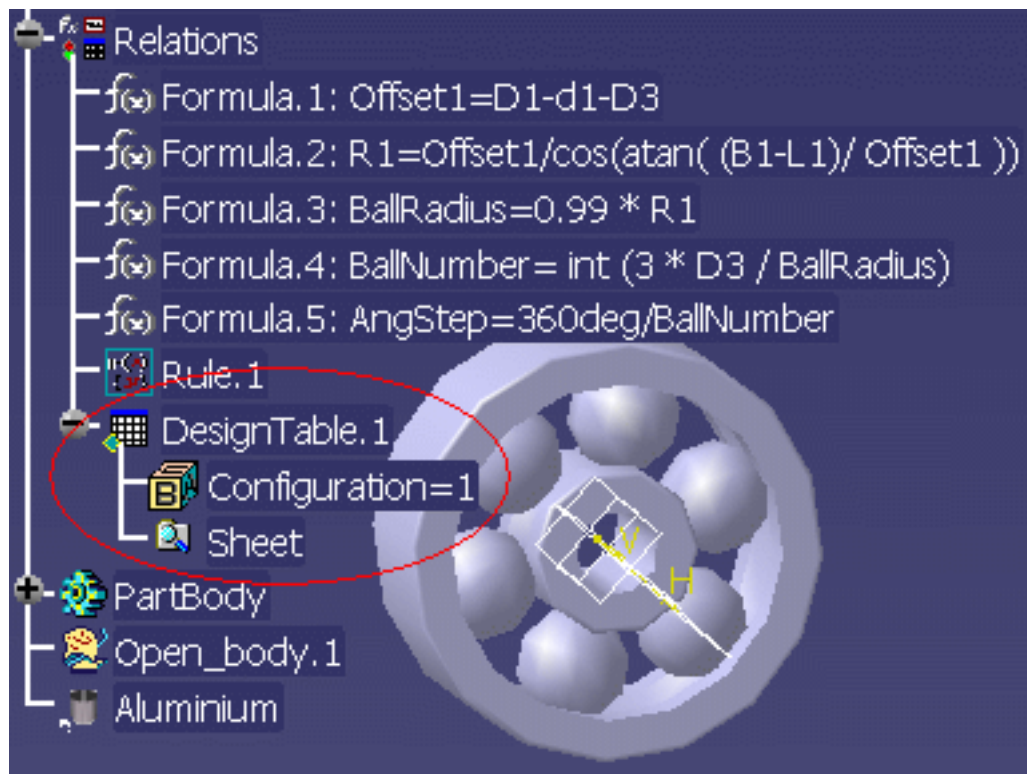
3. Click the Design Table icon (). The Creation of a Design Table dialog box displays.
4. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.
5. Select the [KwrBearingDesignTable.xls](#) file and click **Open**. Click **Yes** when asked if you want to associate the columns of the tables with the parameters.
6. Click **OK** to apply the default configuration.
7. Save your file and close it.
8. Go to the directory containing the KwrBearingDesignTable.xls file and delete it.
9. Go back to Catia. Open the KwrBallBearing1.CATPart file: A dialog box displays asking you if you want to select a new file. Click the **Select** button and select a new Excel file.

Manual Synchronization



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

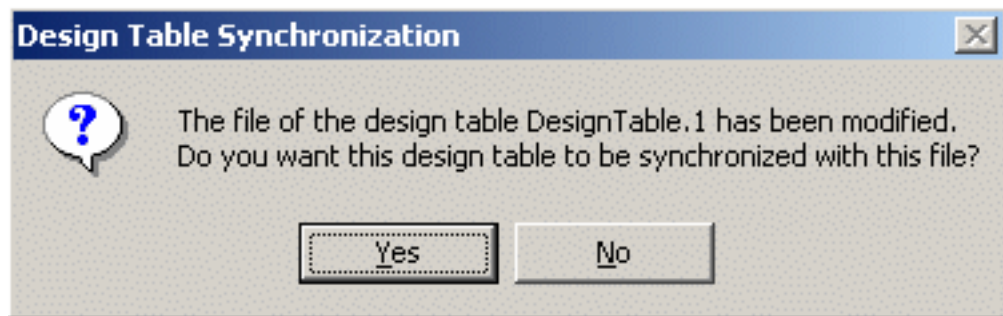
1. From the **Tools->Options...** menu, select **General->Parameters and Measure** and check the **Manual Synchronization At Load** in the Knowledge tab.
2. Open the [KwrBallBearing2.CATPart](#) file. This file already contains a design table whose values are identical to those contained in the [KwrBearingDesignTable.xls](#) file (Note that the [KwrBearingDesignTable.xls](#) file and the [KwrBallBearing2.CATPart](#) file should be located in the same directory.)



3. Select the **Edit->Links** command to edit the Excel file path and select the appropriate [KwrBearingDesignTable.xls](#) file. Save the file and close it.
4. Open the [KwrBearingDesignTable.xls](#) file and modify the material values for example. Close the file.
5. Go back to *CATIA*. Open the [KwrBallBearing2.CATPart](#) file.
6. Select the **Edit->Links** command and click the Synchronize button to synchronize both files.



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



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



Storing a Design Table in a PowerCopy



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

This scenario is divided into the following steps:

- [Inserting the Design Table into the CATPart file](#)
- [Creating the PowerCopy](#)
- [Instantiating the PowerCopy containing the Design Table](#)



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



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

- [KwrBallBearing1.CATPart](#)
- [KwrBearingDesignTable.xls](#)




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

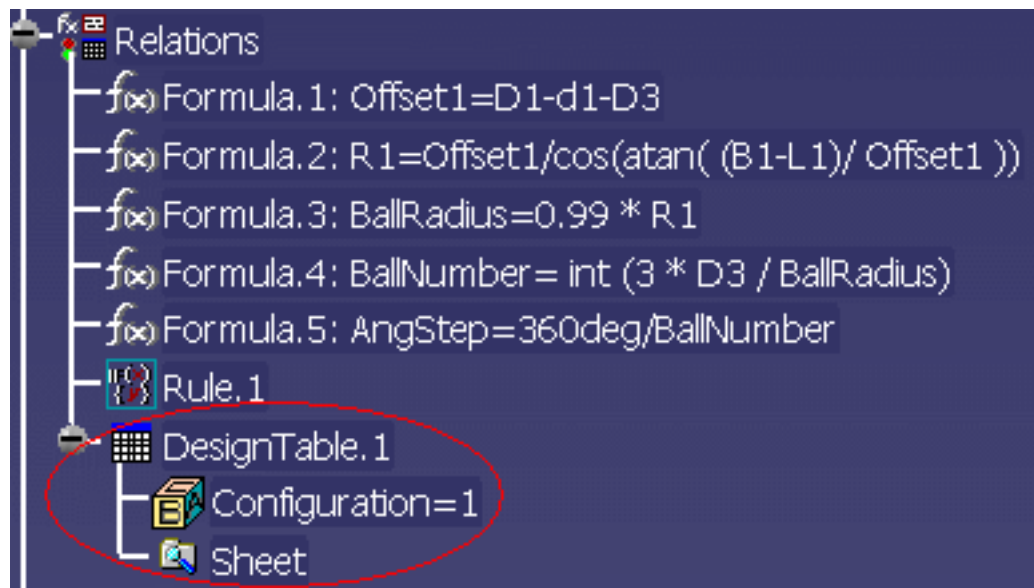


1. Open the [KwrBallBearing1.CATPart](#) file. The following image displays.



Inserting the Design Table into the CATPart file

2. Click the **Design Table** icon () in the Standard toolbar. The **Creation of a Design Table** dialog box displays.
3. Check the **Create a design table from a pre-existing file** radio button and click **OK**. The **File Selection** dialog box displays.
4. Select the [KwrBearingDesignTable.xls](#) and click **Open**.
5. Click **Yes** when asked for automatic associations and click **OK**. The Design table now displays below the Relations node.



Creating the PowerCopy

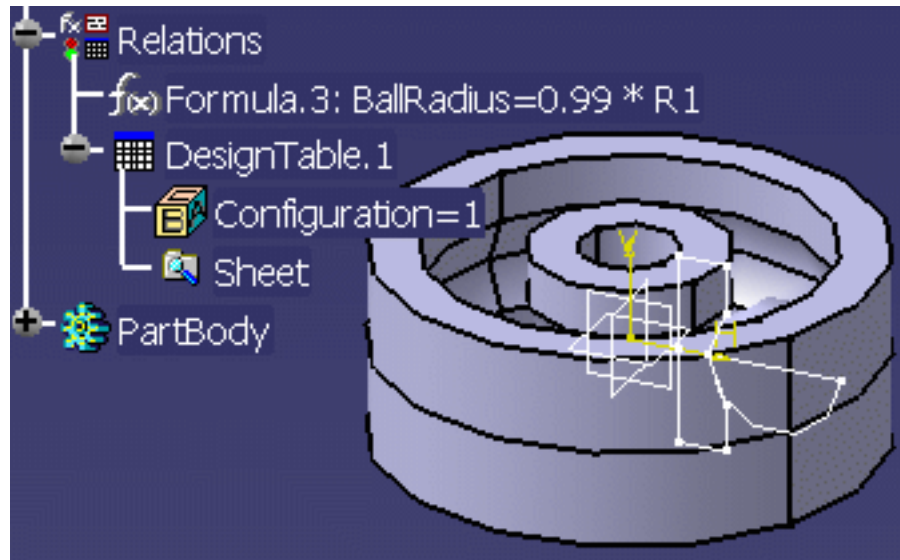
6. From the **Start->KnowledgeWare** menu, access the **Product Knowledge Template** workbench (if need be) and click the **Create a PowerCopy** icon. The Powercopy Definition dialog box displays.
7. In the Specification tree, select the following items:
 - DesignTable.1
 - Shaft.1
 - Shaft.2
 - Shaft.3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



8. Save your file and close it.

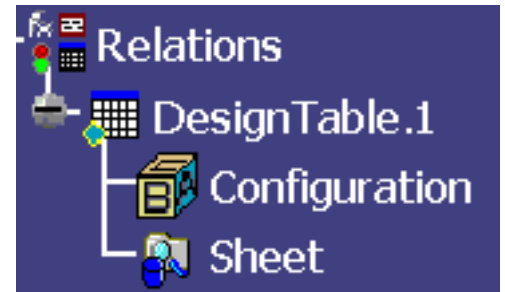
Instantiating the PowerCopy

9. From the **File->New** menu, select **Part** in the **List of Types** and click **OK**.
10. If need be, from the **Start->Knowledgeware** menu, access the **Product Knowledge Template** workbench and click the **Instantiate From Document** icon. The **File Selection** dialog box displays.
11. Select the **KwrBallBearing1.CATPart** file and click **Open**. The **Insert Object** dialog box displays.
12. Select the **yz plane** in the specification tree and click **OK**. The Design Table is instantiated



Working with Design Tables in ENOVIA LCA

- The external files (.xls, or .txt files) of the design tables that can be seen when using the **Edit->Links** command are embedded when saving the file in ENOVIA LCA. Note that this file does not display in the list of the saved files in the **Save in Enovia V5** window but it does in the progression bar when saving the file.
- In a session, if the file containing the design table is closed, the link with the design table external file is not lost i.e. if you re-open the file, the same design table external file will be used. (This allows users to share external files.)
- When saving the sheet in Enovia, the **Duplicate data in CATIA model** option is automatically enabled. If the file is re-opened in another session, the model saved is loaded in session, but the sheet is not.
To synchronize the design table, right-click it, and select the **Synchronize (Enovia)** command. If the design table icon is red, it means that the design table is not synchronized, if it is green, it is synchronized, if it is orange, it is disconnected so you cannot determine if it is synchronized or not. To get an example, see [Saving a Design Table in ENOVIA LCA](#).
- Note that when working with design tables managed by other applications, like Product Engineering Optimizer for example, the **Duplicate data in CATIA model** option will not be automatically enabled when saving the document in ENOVIA LCA (the model size can be too large or the design table file contains too much data): The file will only be projected if CATIA requires it and not if the user wants it.
- Note that if you want to modify/update a design table saved in ENOVIA LCA in CATIA, you will have to check it out in ENOVIA LCA first.
- **Note that the design table features cannot be saved in a product saved in Structure Exposed storage mode - Document not kept.**
- It is possible to create a new version of a design table in ENOVIA LCA. To know more, see [Versioning a Design Table](#).
- Dependency is supported in ENOVIA LCA: Using the **Impacted By** and **Impact On** commands, the user can determine which Part is attached to a design table. To know more about these commands, see the Impacts On and Impacted By topics in the *Engineering LifeCycle User's Guide*.



- Naming the design table: 2 design tables cannot have the same name. For existing models, it is highly recommended to use the **ExportContentToFile** command in ENOVIA LCA: The link will be replaced in case of name conflict. In case of name conflict, you will not be able to save your design table in your database.

[Saving a Design Table in Enovia LCA](#)
[Using a Design Table Saved in ENOVIA LCA in another Part](#)
[Versioning a Design Table External File](#)

Saving a Design Table in ENOVIA LCA



This task is designed to show the user how to save a design table and its external file in ENOVIA LCA. It is divided into the following steps:


- In CATIA, the user creates the design table from a pre-existing file.
- He saves the file in ENOVIA LCA.
- He renames the design table external file so that it is not available for the application.
- He performs a Search in the ENOVIA LCA database.
- The file is sent from ENOVIA LCA to CATIA and synchronized with the Enovia database.



To know more about the use of design tables in ENOVIA LCA, see [Working with Design Tables in ENOVIA LCA](#).



Creating the design table

1. Open the [BallBearing.CATPart](#) file.
2. Click the **Design Table** icon () in the Knowledge tool bar. The **Creation of a Design Table** window displays.
3. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.
4. Select the [BearingDesignTable.xls](#) file and click **Open**. Click **Yes** when asked if you want to associate the columns of the table with the parameters.
5. Click **OK** to apply the default configuration. DesignTable.1 displays below the Relations node.

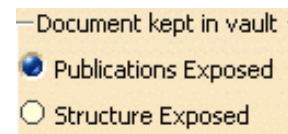
Saving the file in ENOVIA LCA

6. Save your files in ENOVIA LCA. To do so, proceed as follows:

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

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

- Select both files and check the **Publications Exposed** storage mode.




- Check the **Immediate Commit** check box (if need be) and click **OK** to validate. Close the file in CATIA.

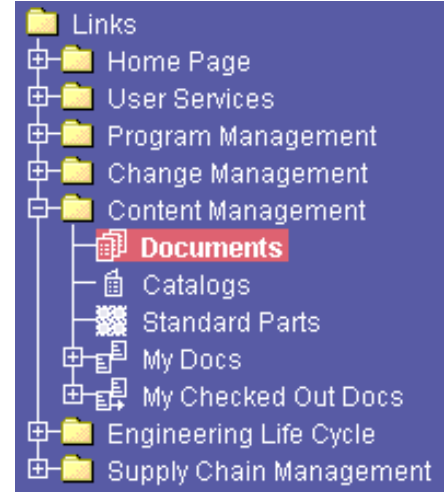


Note that the .xls file does not display in this window. It will only display in the progression bar when saving the data in ENOVIA LCA.

- In Windows Explorer, rename or delete the BearingDesignTable.xls file.

Searching for the saved design table in ENOVIA LCA

- In ENOVIA LCA, click the **ENOVIA Home** icon (). Expand the Content Management node and double-click the Documents folder.



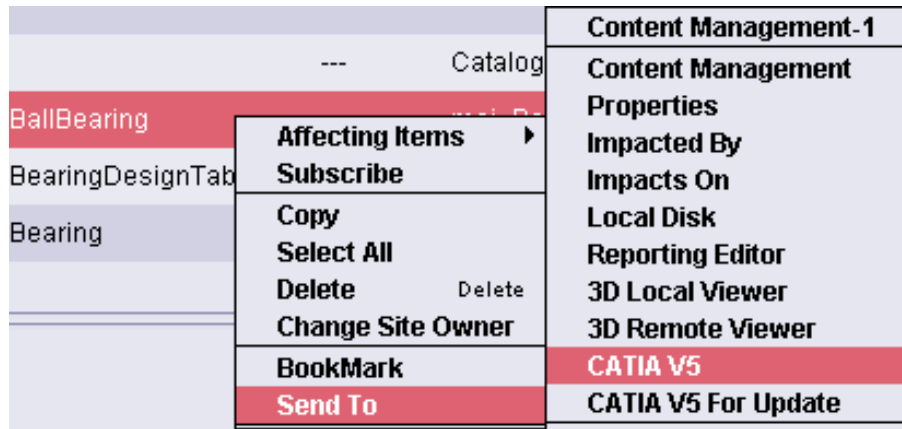
- The Content Management Startup Selection dialog box displays. Click the **Search Documents** radio button and click **OK** to launch the search.



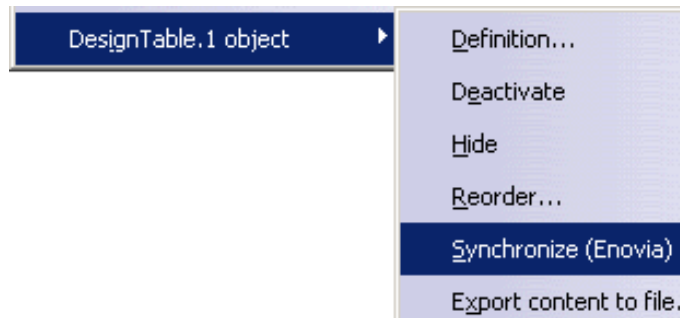
- In the opening Search window,
 - select **Document Revision** in the Search for: scrolling list.
 - select the Creator field, enter the Creator's name and click **Search**. The BallBearing.CATPart file displays in the Results tab.

Sending the File from ENOVIA LCA to CATIA

- Right-click the BallBearing.CATPart file and select the **Send to CATIA V5** command.



- Expand the relations node. Right-click the DesignTable.1 file and select the **Synchronize (Enovia)** command.



The DesignTable.1 displays with a green icon (🟢) indicating that it is synchronized.



Using a Design Table Saved in ENOVIA LCA in another Part



This task is designed to show the user how to use a design table saved in ENOVIA LCA in a new Part. It is divided into the following steps. The user:

- Saves the .xls file in ENOVIA LCA.
- Opens the Part that will contain the .xls file in CATIA and saves it in ENOVIA LCA.
- Performs a Search in the ENOVIA LCA database.
- Sends the founded Part to CATIA.
- Saves the design table .xls file on a local disk.
- Creates the design table file in CATIA and saves the .CATPart file in ENOVIA LCA.
- Sends the file from ENOVIA LCA to CATIA and performs a synchronization.



To know more about the use of design tables in ENOVIA LCA, see [Working with Design Tables in ENOVIA LCA](#).




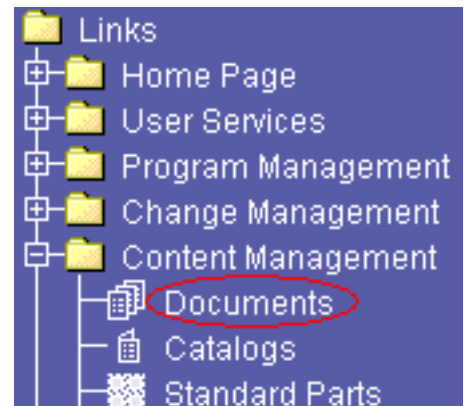
Saving the .xls file in ENOVIA LCA

1. Open your ENOVIA LCA database.

2. In ENOVIA LCA, click the **ENOVIA Home**

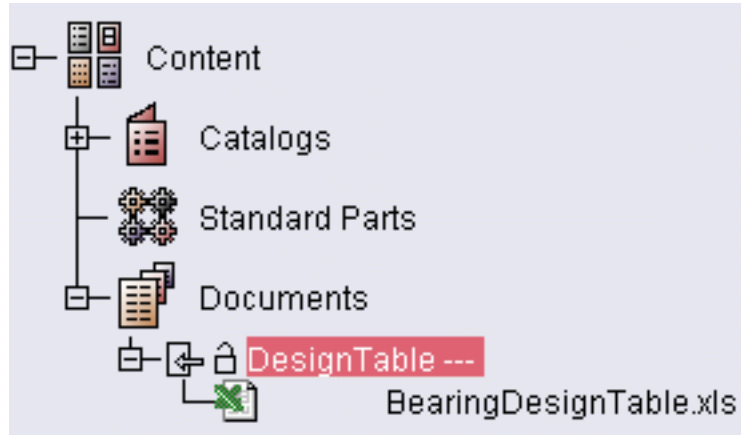


icon (). Expand the Content Management node and double-click the Documents folder.




3. The Content Management Startup Selection dialog box displays. Click the **Create a New Document** radio button and click **OK**.
4. In the **Document ID** field of the opening panel, enter New_Designtable and click the **Browse** button. The Choose a file dialog box displays.

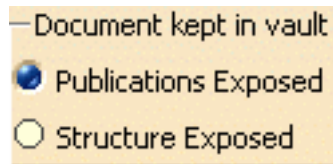
5. Select the [BearingDesignTable.xls](#) file and click **OK**. The design table external file is saved in your ENOVIA LCA database.



6. In CATIA, open the [BallBearing.CATPart](#) file.

7. Click the **Connect to ENOVIA LCA** icon () to connect your ENOVIA LCA database.

8. Click the **Save data in ENOVIA LCA Server...** icon () . The Save in ENOVIA V5 dialog box displays. Select the file and check the **Publications Exposed** storage mode.



9. Check the **Immediate Commit** check box (if need be) and click **OK** to validate. Close the file in CATIA.

The .xls file and the new .CATPart file are now both saved in ENOVIA LCA.

8. In ENOVIA LCA, expand the Content Management node (if need be) and double-click the Documents folder.
9. The Content Management Startup Selection dialog box displays. Click the **Search Documents** radio button and click **OK** to launch the search.

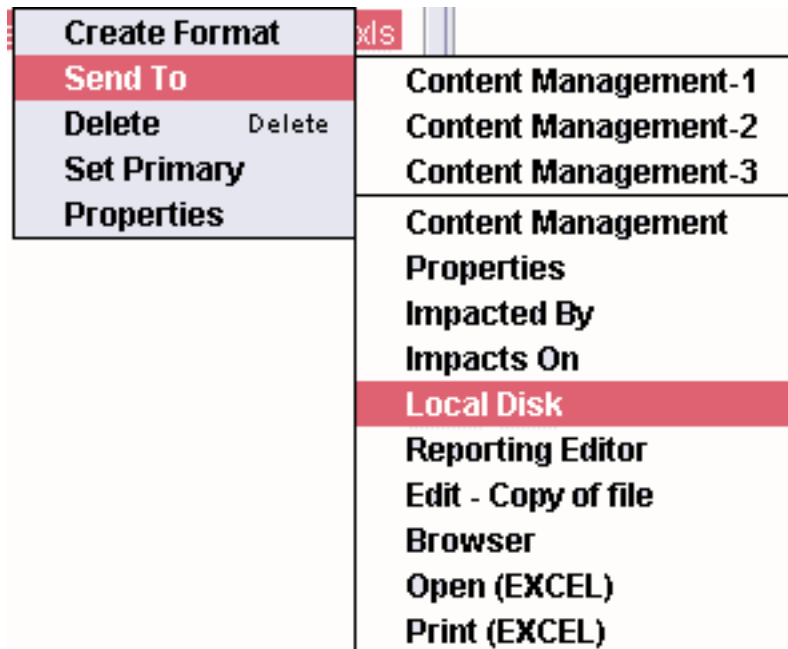


10. In the opening Search window,

- o select **Document Revision** in the Search for scrolling list.
- o select the **Creator** field, enter the Creator's name and click **Search**. The BallBearing.CATPart file and the New_DesignTable display in the Results tab.



11. Right-click the BallBearing.CATPart file and select the **Send To->CATIA V5** command. The file displays in CATIA.

12. In ENOVIA LCA, right-click the New_DesignTable.xls file and select the **Send To->Local Disk** command.



13. In the Copy Out dialog box, indicate the directory where the .xls file will be

stored as well as the file name (BearingDT.xls in this scenario).

14. Click **OK** when done. An information dialog box indicates that the file was correctly saved on your disk. Click **OK** to exit this dialog box.
15. In CATIA, change the part number of the BallBearing.CATPart file into BallBearing2.
16. Click the **Design Table** icon () in the Knowledge tool bar. The **Creation of a Design Table** window displays.
17. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.
18. Select the BearingDT.xls file and click **Open**. Click **Yes** when asked if you want to associate the columns of the table with the parameters.
19. Click **OK** to apply the default configuration. DesignTable.1 displays below the Relations node. Save your file.
20. Click the **Save data in ENOVIA LCA Server...** icon () . The Save in ENOVIA V5 dialog box displays.
21. Select both files, check the **Publications Exposed** storage mode, and the **Immediate Commit** check box (if need be).
22. Close the file in CATIA.

Searching for the saved design table in ENOVIA LCA

23. In the Search tab, click **Search** to launch the search. Your files display in the Results tab.
24. Right-click the BallBearing2.CATPart file and select the **Send To->CATIA V5** command.
25. Right-click the DesignTable.1 file and select the **Synchronize (ENOVIA)** command. The DesignTable.1 is synchronized.



Versioning a Design Table




This task is designed to show the user how to create a revision of a design table external file in ENOVIA LCA.





To know more about the use of design tables in ENOVIA LCA, see [Working with Design Tables in ENOVIA LCA](#).



Creating the design table

1. Open the [BallBearing.CATPart](#) file in CATIA.
2. Click the **Design Table** icon () in the Knowledge tool bar. The **Creation of a Design Table** window displays.
3. Click the **Create a design table from a pre-existing file** option and click **OK**. The File Selection dialog box opens.
4. Select the [BearingDesignTable.xls](#) file and click **Open**. Click **Yes** when asked if you want to associate the columns of the table with the parameters.
5. Click **OK** to apply the default configuration. DesignTable.1 displays below the Relations node.

Saving the file in ENOVIA LCA

6. Save your files in ENOVIA LCA. To do so, proceed as follows:
 - In CATIA, click the **Connect to ENOVIA LCA** icon () to connect your ENOVIA LCA database.
 - In Catia, click the **Save data in ENOVIA LCA Server...** icon () . The Save in ENOVIA V5 dialog box displays.
 - Select both files and check the **Publications Exposed** storage mode.

- Check the **Immediate Commit** check box (if need be) and click **OK** to validate. Close the file in CATIA. Click **OK**: Your data are saved in the ENOVIA LCA database.




Note that the .xls file does not display in this window. It will only display in the progression bar when saving the data in ENOVIA LCA.

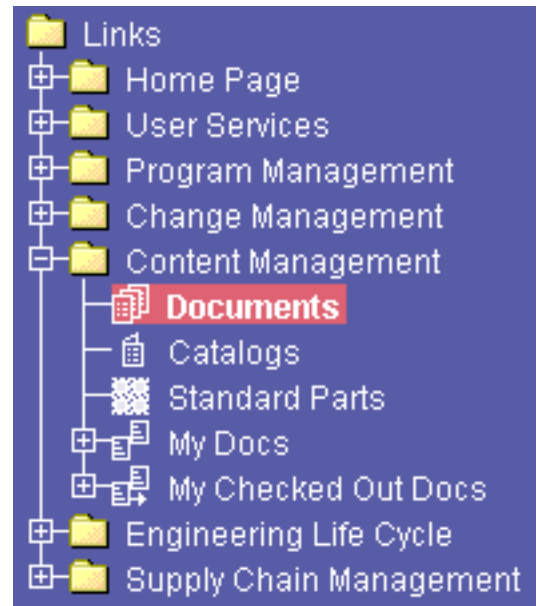
- Close CATIA and relaunch it.

Searching for the saved design table in ENOVIA LCA

7. In ENOVIA LCA, click the **ENOVIA**



Home icon (). Expand the **Content Management** node and double-click the **Documents** folder.



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

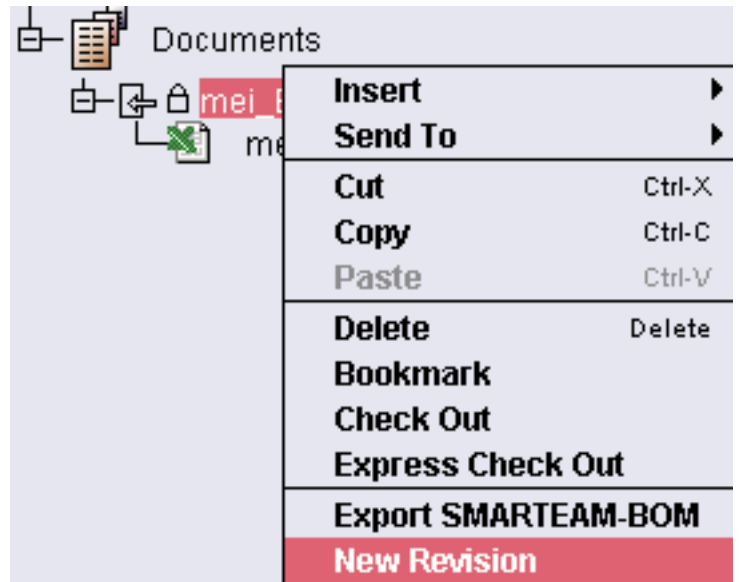


9. In the opening Search window,

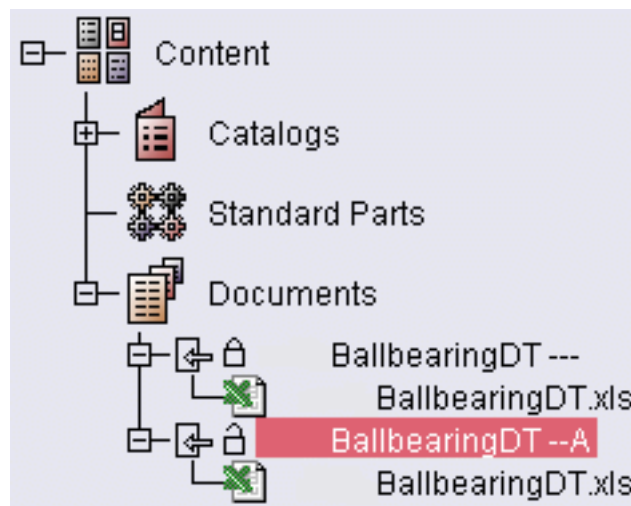
- o select **Document Revision** in the Search for: scrolling list.
- o select the **Creator** field, enter the creator's name and click **Search**. The BallBearing.CATPart file displays in the Results tab.

10. Click the BallBearingDT.xls file and click the **Add** button.

11. Right-click the design table and select the **New Revision** command.



The new revision of the design table is created. Note that the suffix --A is added at the end of the name, indicating that this design table is the first revision of the BallbearingDT design table.



Searching for the saved design table in ENOVIA LCA

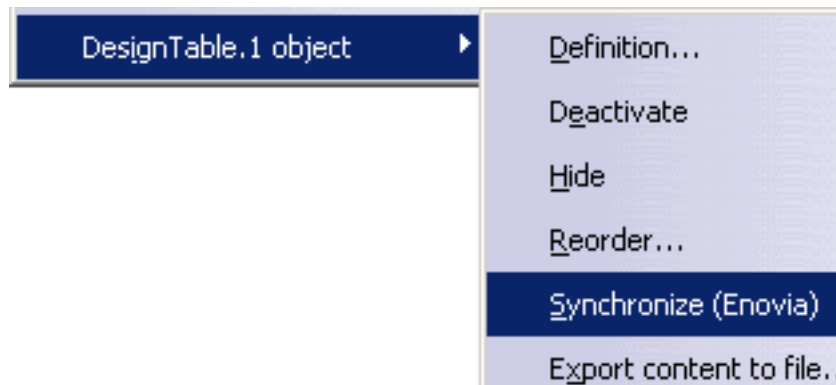
12. Double-click the Documents folder. The Content Management Startup Selection dialog box displays.

13. Click the **Search Documents** radio button and click **OK** to launch the search.

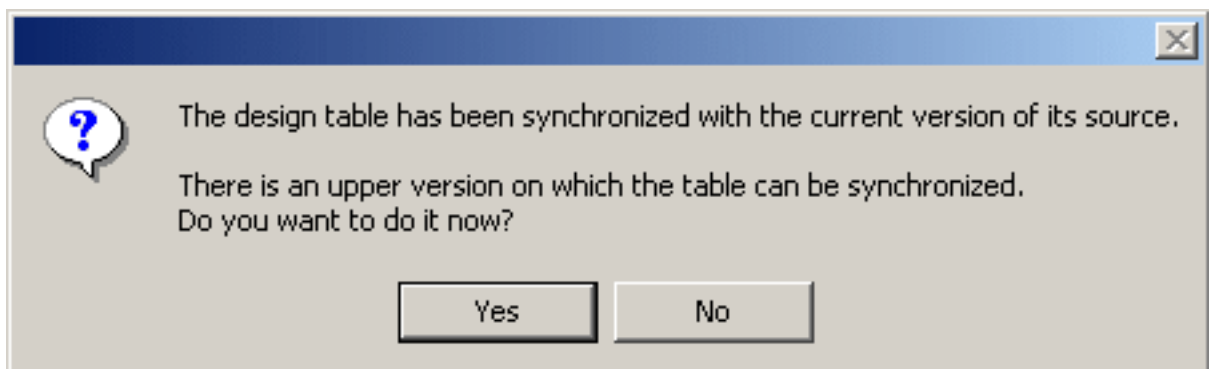
14. In the opening Search window, select
 - select **Document Revision** in the Search for scrolling list
 - select the **Creator** field, enter the creator's name and click **Search**. The BallBearing.CATPart file displays in the Results tab

Sending the File from ENOVIA LCA to CATIA

15. Right-click the BallBearing.CATPart file and select the **Send to->CATIA V5** command.
16. In CATIA, expand the Relations node: The DesignTable.1 displays with a red icon indicating that it is not synchronized with ENOVIA LCA.
17. Right-click the DesignTable.1 file and select the **Synchronize (ENOVIA)** command.

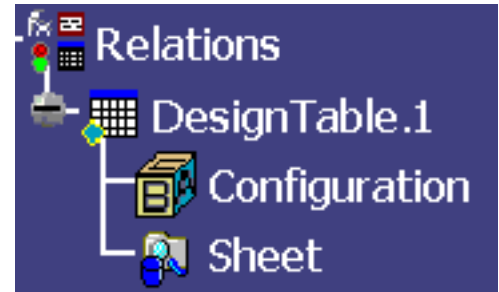


A message displays indicating that another version of the design table is available.



18. Click **Yes** to synchronize the design table with the latest version stored in ENOVIA LCA.

The DesignTable.1 displays with a green icon (🟢) indicating that it is synchronized.



Design Tables: Useful Tips

- A design table can only be created from *non-constrained parameters*, i.e. from parameters which are neither referred to in an active design table nor used in any other *active relation*. If you keep the Activity option checked for DesignTable0 and you try to create another design table, you will have to select the parameters to add to your second design table among a restricted parameter list. Uncheck the Activity option if you want to deactivate a design table and reuse its parameters in another design table.
- Anytime you modify a design table, the relations that refer to this design table detect the modification and turn to a to-be-updated status.
- As long as a design table is active, the parameters which are declared in it are constrained parameters and you are not allowed to modify them. Double-clicking a design table in the specification tree displays the design table with its set of configurations and allows you to select a new configuration.
- Only parameters which are not already constrained by any other relation or by any other design table can be used to create a design table. If a parameter is already constrained, it does not appear in the Parameters to insert list in the design table dialog box.
- **Selecting the parameters to be inserted in a design table**

The Filter Name and Filter Type filters can be used to restrict the display of a parameter list. If you specify x in the Filter Name field of the Select parameters to insert dialog box, you will display all the parameters with the letter x in their name (xA, xB, xC, xD, xE). If you select the Renamed Parameters in the Filter Type list, you will display all the parameters you have renamed in the Formulas dialog box (yA, xB, xA, yC, xC, yB, yD, xD, yE, xE, TangE).

Parameters to be inserted can be multi-selected. You just have to keep on pressing the Ctrl key while you select parameters. If you do this, the group of multi-selected parameters will be carried forward onto the Inserted parameters list in the order in which they are displayed in the initial list. When the design table is created, the rank of the columns fits the rank of the parameters in the Inserted parameters list. If you want to have columns ordered in a given way in the design table, you must insert the parameters one by one.
- **Accessing the functions related to the design table**

Once in the formula (rule or check) editor, select the Design Table item in the dictionary, the list of the methods that can be applied to a design table is displayed. Select a method, then click F1 to display the associated documentation.

The Knowledgeware Language

The documentation of the objects to be manipulated in formulas, rules and checks (for those of you using the Knowledge Advisor product) can now be accessed by clicking F1 in the $f(x)$ dictionary. Just select an item (for example the MaxInColumn method in the Design Table package), then click F1 to display the documentation of the MaxInColumn method.

The knowledgeware language is described in the *Knowledge Advisor User's Guide*.

Working Through the Knowledgeware Capabilities

Introduction - The Design Intent

Calculating and Checking a Volume

Working with a Design Table

About Rule Firing

Knowledgeware Automation

Optimizing a Volume

Appendix: Creating a Deformable Revolution Body

Introduction: Design Intent



Configuration 1
of [KwrProfilesDesignTable.xls](#)
applied to [KwrBottleProfiles.CATPart](#)

The bottle above is used to demonstrate the major knowledgeware techniques that can be used in CATIA Version 5 to help you design a product. Throughout this section, most facets of the CATIA knowledgeware capabilities are examined, from relations such as *formulas* and *rules* to *optimization algorithms*. A scenario is developed in every chapter around a specific theme, and for each scenario tips and techniques are given.

This part is intended for advanced users. Before you tackle the scenarios defined in this guide it is better to have previous knowledge of the products listed below and have an idea of the basic tasks you can carry out with them:

- CATIA Infrastructure
- CATIA Part Design
- CATIA Generative Shape Design
- CATIA Knowledge Advisor

- CATIA Product Optimizer.

Design Intent

When developing a product, you must first of all define your product requirements. These requirements may be the result of mechanical, manufacturing or style considerations. The approach you follow in knowledge-based design consists in integrating these requirements in specific tools so that, for example you can check and validate data during the design process or observe how your document behaves depending on the context. This chapter defines the requirements of the bottle used as an example and explains how to *capture your design intent*, i.e. describe your requirements through knowledge features.

Design Requirements

Suppose you are designing a new bottle and you are required to make proposals to your marketing department. Leaving aside style considerations, you are free to do what you want except for the following restrictions:

- the thickness of the bottle is determined by manufacturing considerations. This is not a parameter you can modify.
- you can propose any bottle shape as long as, for a first estimate, the internal volume remains in the [230 cm³ - 280 cm³] range. This is the only requirement as regards geometrical properties.

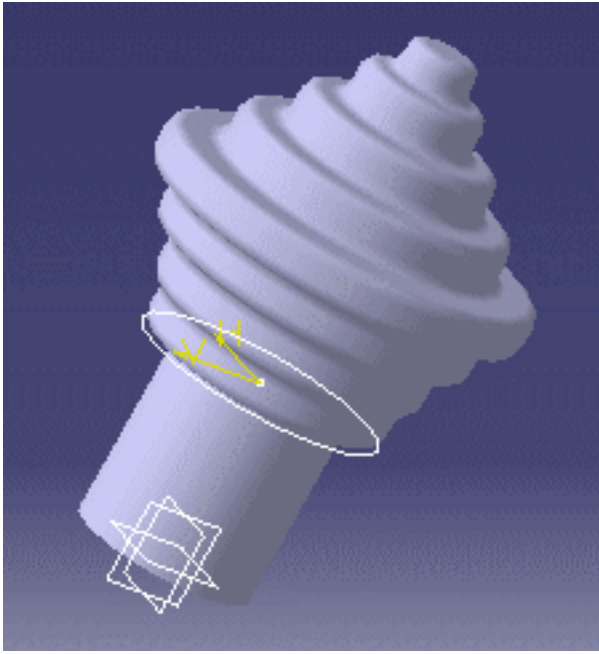
The trick for you now is to determine the volume of the bottle, have this volume updated whenever you change the bottle's shape and be warned should the calculated volume be out of range.

The approach followed to capture this design intent relies on *measures*. The "*measure*" capabilities provide you with a function which calculates the volume of a body. There is no real means to prevent you from designing bottles that are too small or too large, but using *Knowledge Advisor checks* is a good way to be warned whenever the volume is out of range. This is explained in [Calculating and Checking a Volume](#).

Having on hand a series of bottles' shapes fulfilling all requirements, you are required to be able to refine your result and search for a design so that the exact volume of the bottle is 250 cm³. To achieve this goal, we use the *Product Synthesis Optimizer* capabilities and both available algorithms to design the final bottle shape. This is explained in [Optimizing a Volume](#).

Assembly Requirements

It is planned to provide the bottle with a cap. After reviewing development plans with the marketing department, it has been decided that selling this new perfume brand with an already existing cap would be a saving. The cap they plan to reuse is the one below:



Reusing this cap affects the bottle neck design which must allow for a certain section as well as a certain depth.

Although it is planned to re-design this cap, the marketing department would like to have an idea of the assembly made up of the bottle and the cap. To check the overall design of the product, the marketing people want to be able to generate the bottle assembly automatically on screen each time a bottle shape meets the requirements.

A CATIA Knowledgeware answer to this problem is a Knowledge Advisor rule which can be triggered whenever certain conditions are fulfilled and generate automatically the global assembly from a VB macro. This is explained in detail in [About Rule Firing](#). How you record , replay or modify a macro is discussed in [CATIA Knowledgeware Automation](#).

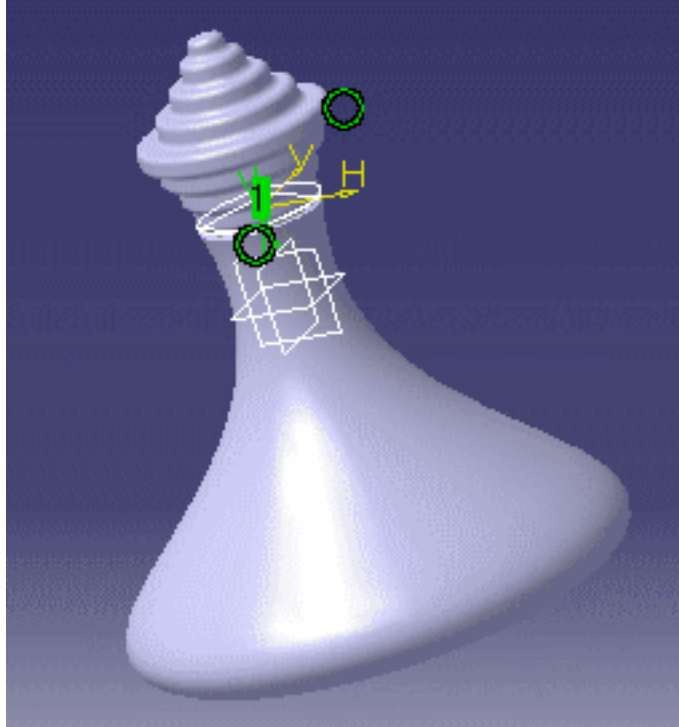
Style Requirements

The Marketing Department already has an idea about the shapes they would like to study. The proposed shapes should be revolution bodies. The longitudinal view of the product should exhibit no edges. In other words, all contours are intended to be smooth. Stylists want you to provide them with a flexible design, they want profiles that are easy to be deformed by controlling one or more parameters. They want an immediate result on screen and they also want a clue as to whether they are still working within the authorized limits.

Our initial feature is a five-point spline. The points making up the bottle neck are fixed as they must accommodate the cap. The other points are those providing the required flexibility. You alter them to create a new bottle shape. To create a bottle, you must create the spline, then rotate this spline. At this stage you obtain a revolution surface open at both ends. Then you have to create a *fill* at the aperture which is assumed to be the bottom of the bottle, join all the faces and thicken the resulting joined surface. All this is described in [Appendix: Creating a Deformable Revolution Body](#).

After a new shape has been designed and satisfies the requirements, you can store its parameters in a design table. The design table is a way to gather in an external file all the profiles satisfying the requirements. How to create a design table storing the data of all profiles and how to use a design table are explained in [Working with a Design Table](#).

Calculating and Checking a Volume



Configuration 2
of [KwrProfilesDesignTable.xls](#)
applied to [KwrBottleProfiles.CATPart](#)

The bottle we start from is described in [Appendix: Creating a Deformable Revolution Body](#). First of all, we want to be able to measure the bottle's volume, then each time the bottle's profile is modified, we want to be warned about whether the resulting volume is still in the [230 cm³ - 280cm³] range. To achieve this goal, we will be using two CATIA knowledgeware capabilities, the *measures* and the *checks*. Measures are functions provided by various applications such as Part Design or Generative Shape Design to compute data. These functions can be used in formulas as well as in rules and checks. They can be accessed from an interactive dictionary. Checks are relations that don't modify the document but just tell you whether certain specified criteria are fulfilled.


Specifying the Proper Settings

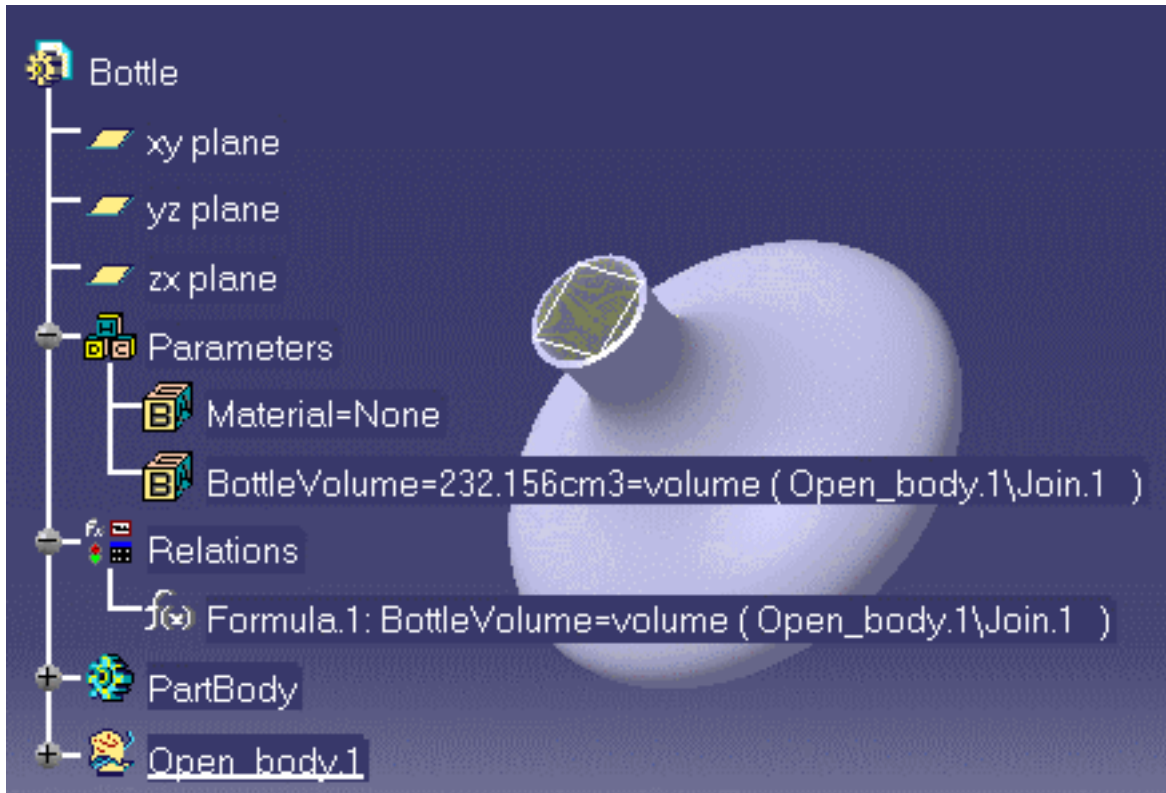
Before going any further in the scenario developed in this Part, check the settings below:

- In **Tools->Options->General->Parameters and Measure**.
- In the **Language** tab, the **Load extended language libraries** box must be checked otherwise you won't be able to access the **Measures** in the knowledgeware dictionary.
- In the **Knowledge** tab, check the **With Value and With Formula** check boxes.
- In the **Units** tab, specify cm³ as the default volume unit.

- In **Tools->Options->Infrastructure->Part Infrastructure->Display**, check at least the **Relations** and **Parameters** boxes. But it is recommended to check all the options below the specification tree settings.

Calculating the Bottle Volume


1. Open the [KwrThickSurface.CATPart](#) document.
2. Click the  icon or select the **Tools->Formula** command from the standard menu bar. The "Formula" dialog box is displayed.
3. Select the **Volume** item in the **New Parameter of type** list. Then click **New Parameter of type**. A parameter called **Volume.1** is displayed and highlighted in the parameters list.
4. In the **Edit name or value** of the current parameter field, replace the **Volume.1** name with **BottleVolume**.
5. In the dictionary, select the **Measures** item, then double-click **Volume** in the measure list. If need be, add parentheses after the function name in the formula editor. At this stage the formula must be:
BottleVolume = volume()
6. Position the cursor between the parentheses and capture the joined surface (**Join.1**) definition from the specification tree. To do this, just double-click the **Join.1** feature. The formula definition you should get in the editor is something like:
BottleVolume = volume (Open_body.1\Join.1)
7. Click **OK** in the Formula Editor. You are back to the **Formulas** dialog box. The new formula is displayed in the parameter list opposite the **BottleVolume** parameter. It is also displayed in the specification tree under the **Parameters** and **Relations** nodes. Click **OK** again in the **Formulas** dialog box to exit the **Formulas** dialog.



Checking the Volume Value

1. Select the document root feature, then access the Knowledge Advisor workbench.

To do this, select the **Start->Knowledgeware->Knowledge Advisor** command from the tool bar.

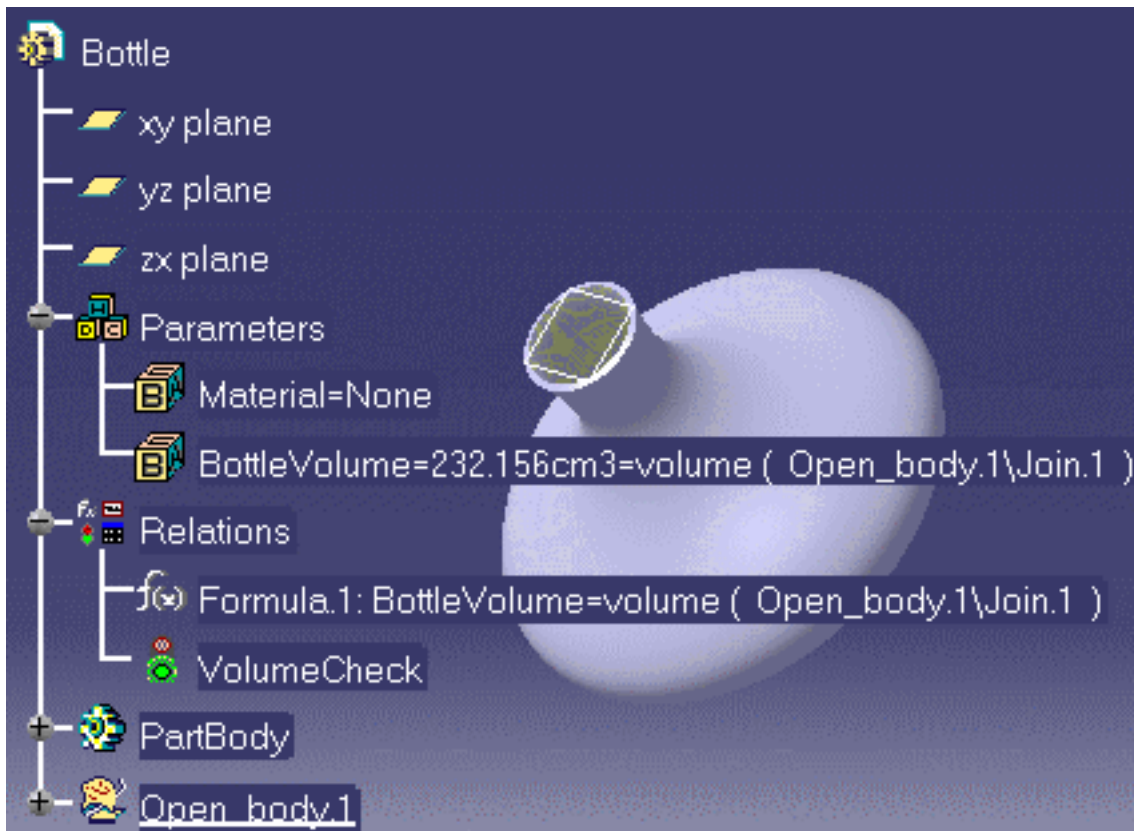
2. Click the  icon to display the Check Editor. In the first dialog box, replace the default name with VolumeCheck. Click **OK**. The Check Editor is displayed.

3. Define your check. To do this:

- a. Select **Information** or **Warning** in the Type of Check list.
- b. Enter the string "Volume out of range" in the message field.
- c. Enter the statement below in the edition window:

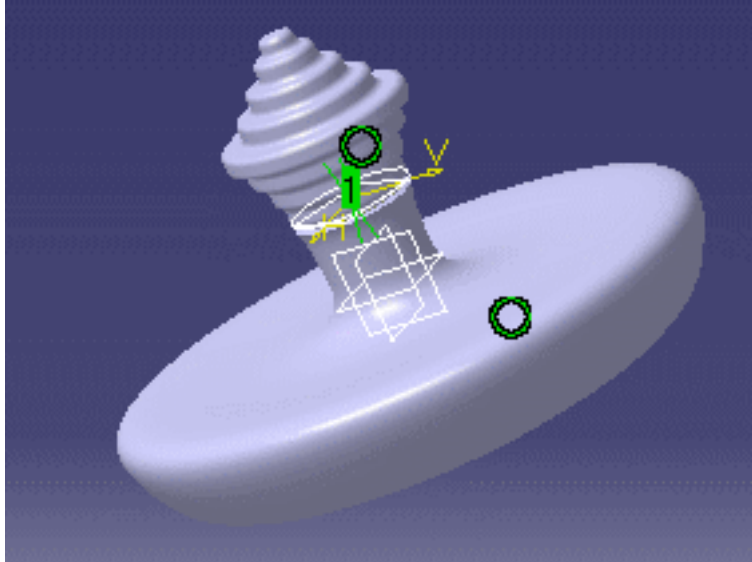
$(\text{BottleVolume} > 230 \text{ cm}^3) \text{ and } (\text{BottleVolume} < 280 \text{ cm}^3)$

4. Click **OK** to exit the dialog box and add the check to the document. In the specification tree, the check icon is green.



The resulting document is [KwrVolumeCheck.CATPart](#).

Working with a Design Table




Configuration 3
of [KwrProfilesDesignTable.xls](#)
applied to [KwrBottleProfiles.CATPart](#)

You now have a preliminary document. How are you going to warp the profile of the bottle and search for other shapes? How can you capture all the data fulfilling the requirements? If need be, how are you going to proceed to refine your design in order to obtain a given volume? Using a design table is a way to capture your design intent and modify your document through an external file. In the scenario below, we first deform the bottle shape interactively by manipulating the control points of the spline, then we create and enrich a design table from the data of the documents fulfilling the requirements.

Searching Interactively for Valid Shapes

1. Open the [KwrVolumeCheck.CATPart](#) document.
2. Double-click the Sketch.1 feature and manipulate the D and E control points to deform the spline. See [Appendix: Creating a Deformable Revolution Body](#) for a definition of the D and E control points. As soon as you deform the spline:
 - a. The document color changes (by default it turns to red).
 - b. In the specification tree, an update icon is displayed on the root feature and on the formula.
3. Update the Part by double-clicking the root feature in the specification tree or in the geometry area.
4. Access the Knowledge Advisor workbench by double-clicking the formula in the specification tree.

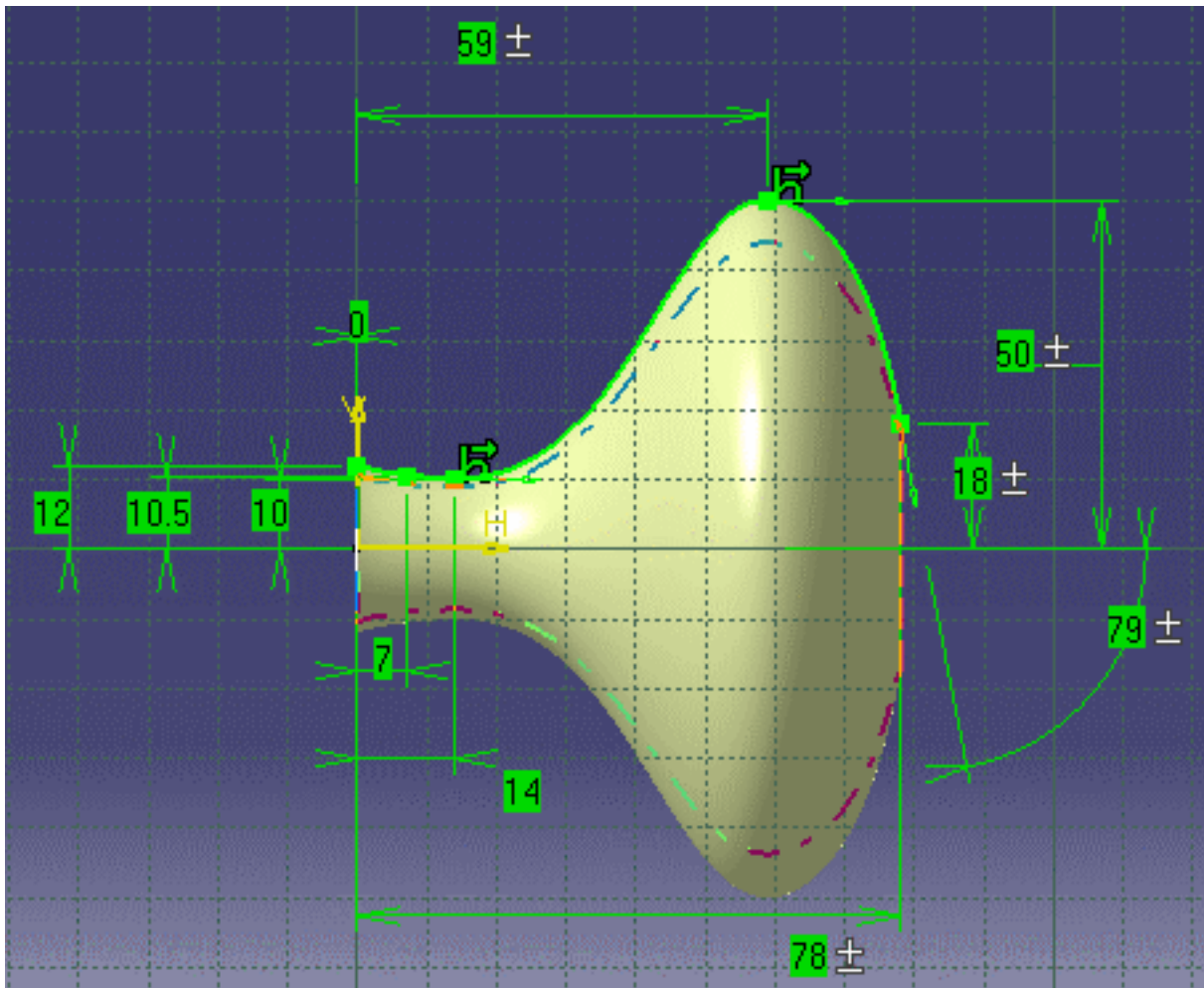


- Click the  icon to update the formula. If the recalculated volume does not satisfy the requirements, a message box is displayed informing you that the volume is out of range and the check icon turns red. Otherwise, the check icon turns green.


Redo these interactions until you obtain a bottle satisfying both your style criteria and your volume requirements. As soon as you find the right profile, follow the method below to create a design table intended to store the data related to this profile as well as the data corresponding to other profiles.

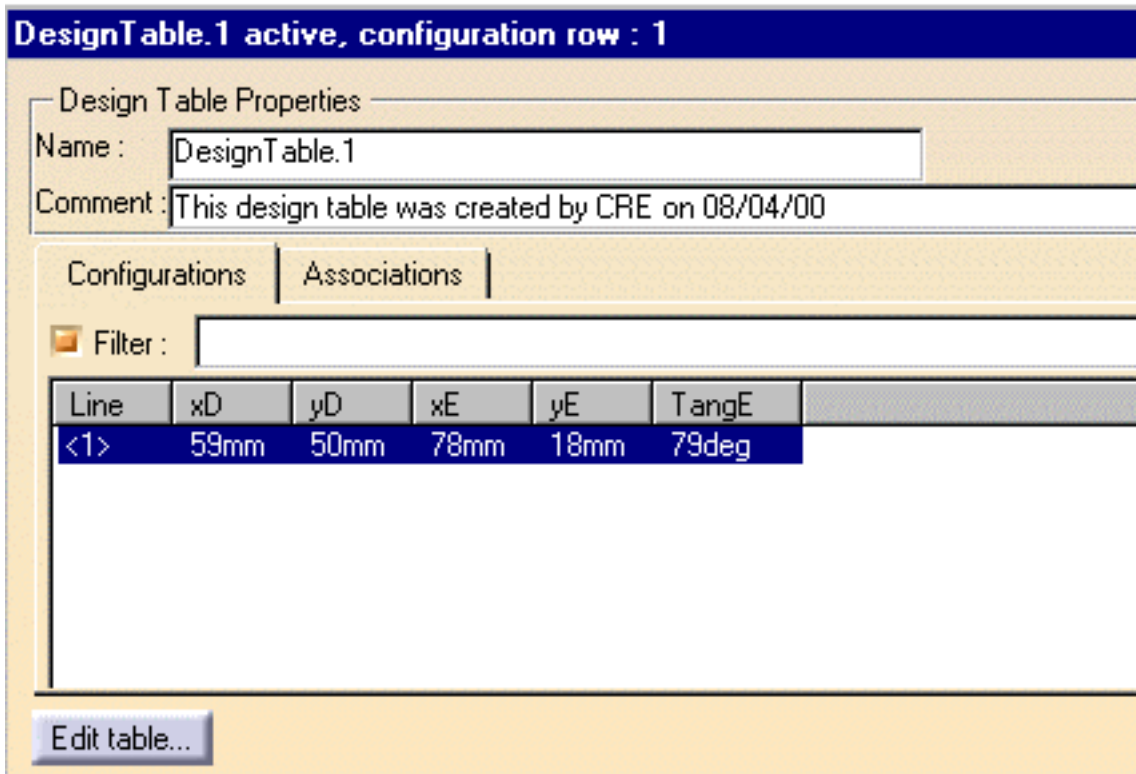
Initializing and Enriching a Design Table

- Open the [KwrVolumeCheck.CATPart](#) document and save it under a new name.
- In the renamed document, access the Sketch.1 feature and add the xD,yD, xE and yE offset constraints on the D and E control points. The constraints named x are defined along H while those named y are along V. Add a tangency constraint on E with respect to H as on the figure below and name it TangE. Refer to [Appendix: Creating a Deformable Revolution Body](#) for the definition of the control points. The figure below is an example of a valid shape.



If need be, update the document and the formula.

3. In the standard toolbar, click the  icon to create a design table.
4. In the Creation of a Design Table dialog box, check the **Create a design table with current parameter values** box. Click **OK**. The **Select parameters to insert** dialog box is displayed.
5. Select the **xD, yD, xE, yE** and **TangE** parameters in the **Parameters to insert** list. Use the right arrow to move them to the **Inserted parameters** list. Click **OK**.
6. In the **Select the Pathname of the File to be Created** dialog box, specify the pathname of an .xls file to store the design table, and click **Open**. A single row design table similar to the one below is displayed.



7. Edit the created design table by clicking the **Edit table...** button. The Microsoft Excel application is started. A single row table with the **xD, yD, xE** and **yE** and **TangE** parameter values is displayed. Click **OK** in the design table dialog box to add the created design table to your document. Save and close this document.
8. Go back to **KwrVolumeCheck.CATPart**, edit the sketch and search for another profile. As soon as you think a profile is worth saving, edit the control points and carry forward the edited values to the Excel table you have just created. Each time you do this, you add a new row to the design table.

- After the data of all the profiles to be stored have been added to the Excel table, close this .xls file and re-open the renamed document. Re-edit the design table, select one by one each design table configuration in order to display the various shapes on screen.

If you enrich the created design table with the data below:

KwrProfilesDesignTable.xls					
	A	B	C	D	E
1	xD (mm)	yD (mm)	xE (mm)	yE (mm)	TangE (deg)
2	59	50	78	18	79
3	82	53	91	26	90
4	29	65	48	47	78
5	57	46	90	21	10
6	72	38	125	15	10
7	56	38	105	10	90

This is what you get on screen when this design table is edited in the renamed document.

DesignTable.1 active, configuration row : 2

Design Table Properties

Name :

Comment :

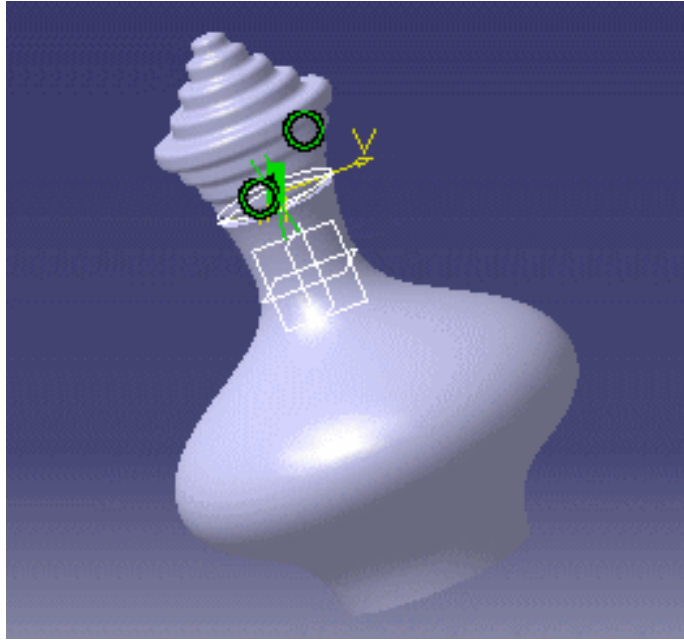
Configurations | Associations

Filter :

Line	xD	yD	xE	yE	TangE
1	59mm	50mm	78mm	18mm	79deg
<2>	82mm	53mm	91mm	26mm	90deg
3	29mm	65mm	48mm	47mm	78deg
4	57mm	46mm	90mm	21mm	10deg
5	72mm	38mm	125mm	15mm	10deg
6	56mm	38mm	105mm	10mm	90deg

The design table above is provided in the KwrProfilesDesignTable.xls sample. The bottle shape corresponding to each configuration of this design table is depicted by the figure which starts each chapter of this part.


About Rule Firing




Configuration 4
of [KwrProfilesDesignTable.xls](#)
applied to [KwrBottleProfiles.CATPart](#)

This scenario explains how to write rules and gives some tips about the process which is behind the rule firing. Two rules are created. One launches a macro which updates and saves the document whenever the design table configuration results in a valid check. The other starts a macro which generates an assembly with constraints.

Writing the Rules

1. Open the [KwrBottleProfiles.CATPart](#) document.
2. Click the  icon. In the dialog box which is displayed, check the Create a design table from a pre-existing file box. Click OK. In the file selection dialog box, select the [KwrProfilesDesignTable.xls](#) sample. Click OK in the box which asks you whether you want to associate automatically columns with parameters. Select Configuration 2 as the active configuration. Save the document under the [KwrTipCreateAssembly.CATPart](#) name.
3. Check that the [KwrTipSave.CATScript](#) and [KwrTipCreateAssembly.CATScript](#) macros are downloaded in your environment as well as the [KwrCap1.CATPart](#) document. In the [KwrTipCreateAssembly.CATScript](#) macro, replace the path defining the assembly components in the `var1(0)` and `var2(0)` definitions.
4. Deactivate the design table from the contextual menu.

5. Access the Knowledge Advisor workbench to create the UpdateAndSaveRule rule. To do this:

- a. Click the  icon.
- b. In the first dialog box which is displayed, enter the UpdateAndSaveRule name. Click OK to display the main rule editor.
- c. In the rule editor, enter the rule below:

```
if Relations.1\DesignTable.1\Configuration < 7
{
  Message("Document update and save")
  LaunchMacroFromFile("e:\users\...\KwrTipSave.CATScript")
}

else Message("Configuration # is invalid",
Relations.1\DesignTable.1\Configuration)
```

Prior to clicking OK, replace the path specified in the LaunchMacroFromFile function with the path where you have downloaded the KwrTipSave.CATScript macro.

- d. Click OK to add the rule to the document. A message box is displayed (Document update and save). This message box is generated by the Message function in the rule. Two Visual Basic boxes are also displayed. They are generated by the KwrTipSave.CATScript macro.
 - e. Deactivate the rule.
6. Use the same procedure to create the AssemblyRule rule below:

```
if Relations.1\UpdateAndSaveRule\Activity == true
LaunchMacroFromFile("e:\...\KwrTipCreateAssembly.CATScript")
```

Prior to clicking OK, replace the path specified in the LaunchMacroFromFile function with the path where you have downloaded the KwrTipCreateAssembly.CATScript macro.

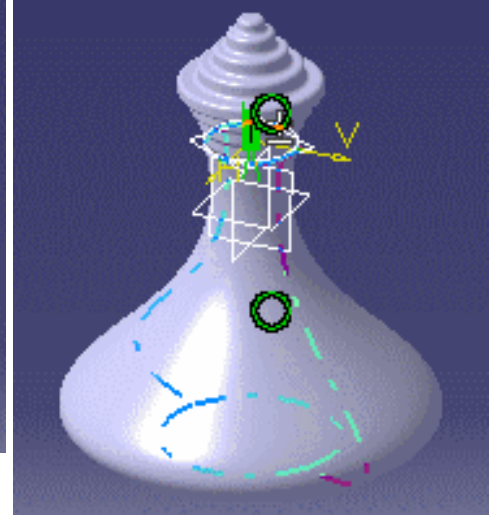
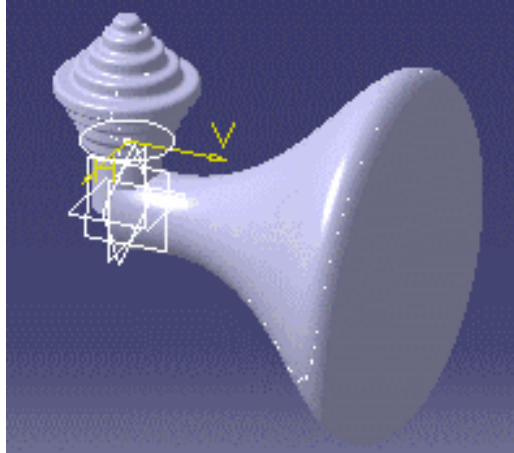
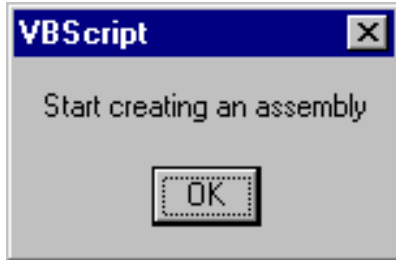
7. Deactivate all the features located below the Relations node of the specification tree then save your document.

Firing the Rules

1. Reactivate UpdateAndSaveRule. The message boxes warning about the update and save operations are displayed.
2. Reactivate DesignTable.1 and modify the design table configuration. To do this, double-click the

design table icon in the specification tree then select a new row among the displayed configurations (from 1 to 6).

3. Reactivate AssemblyRule. The macro which creates the assembly is launched. You get on screen a message warning you that an assembly is going to be created. Click OK in the Visual Basic message box. The assembly is generated. You can see the different steps on screen.



Once the macro has finished running, close the product which has been generated, then deactivate AssemblyRule.

4. Go back to the initial document. Modify again the design table configuration. The UpdateAndSave macro is launched but not the AssemblyRule macro which is deactivated. To create the assembly corresponding to the new configuration, reactivate AssemblyRule.

About Automation



Configuration 5
of [KwrProfilesDesignTable.xls](#)
applied to [KwrBottleProfiles.CATPart](#)

A macro is a way to store instructions intended to be repeated many times. It can also be a good means to store intricate interactions or describe a document in a clear text file easy to edit and requiring less memory than a document. Knowledgeware automation provides you with a way to store operations in the form of a .CATScript file. In this chapter we discuss the macro used to create the assembly in [About Rule Firing](#), then we introduce the knowledgeware automation objects. For more information, see the *Knowledge Advisor Journaling Guide*. We will not dwell on the part which consists in creating an assembly as this guide deals more specifically with knowledgeware techniques. If you need a brush up on how to create an assembly, see the *Assembly User's Guide*.

Creating an Assembly using a CATScript Macro

A CATScript macro is written in a language similar to the Visual Basic language. You can record a CATScript macro, then replay it later on or write it from scratch. The recommended method is to start from a record then, depending on your needs, edit and modify the pre-recorded macro.

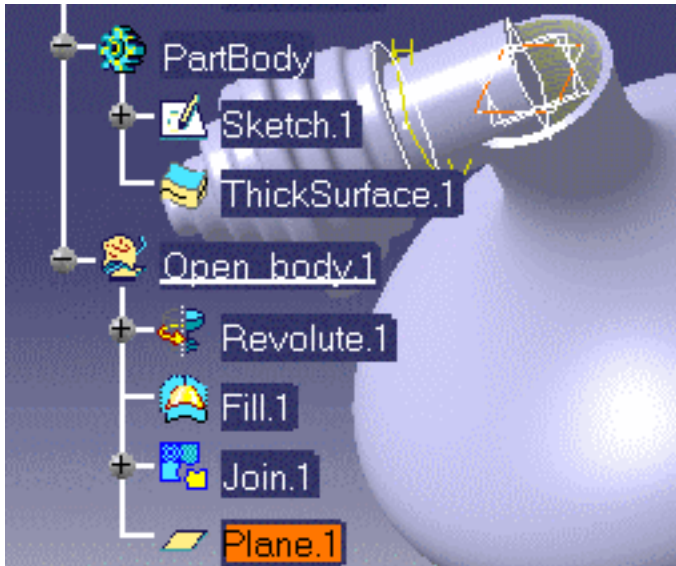
Recording the CATScript Macro

To record the macro used in [About Rule Firing](#):

1. Close all the documents open in your session.
2. Select the **Tools->Macro->Start Recording...** command from the standard menu bar then in the Record Macro dialog box displayed, specify the path of an external file. If need be, see the *CATIA Knowledge Advisor Journaling Guide* for information on how to record a macro. Press the Start button to start the macro recording. From now on, all the interactions are recorded in the CATScript file you have just specified.

Start of recording

- a. Select the **Start->Infrastructure->Product Structure** command from the standard tool bar. The product1 root product is created.
- b. Select the **Components->Existing Component...** command from the root product contextual menu to add the [KwrTipCreateAssembly.CATPart](#) component then the [KwrCap1.CATPart](#) component.
- c. In the Assembly Design workbench, specify an offset constraint of 1 mm between the Plane.1 plane of the bottle and the Pad.2 surface. Refer to the figure below to see how to select the elements to be constrained.



- d. Specify a coincidence constraint to make the cap and the bottle axes coaxial. For more information, refer to the *Assembly User's Guide*. If need be, update the document.

End of recording

3. Select the **Tools->Macro->Stop Recording** command from the standard tool bar. This closes the file which records all the interactions described above.

Taking a Look at the Macro

The CATScript macro you have just recorded is similar to the one below. For the sake of clarity and to make lines shorter:

- Some objects have been declared while others have been renamed
- The arguments passed in functions using the [generic naming](#) method are written in italics.

Before replaying this macro, you should replace the path specified as the argument of the `AddComponentsFromFiles` method.

```
Dim PrDoc0 As Document
Set PrDoc0 = CATIA.Documents.Add("Product")

Dim Prod1 As Product
Set Prod1 = PrDoc0.Product

Dim var1 ( 0 )
' Replace the path below before replaying the macro
var1 ( 0 ) = "E:\...\KwrCap1.CATPart"
Prod1.Products.AddComponentsFromFiles var1, "*"

Dim var2 ( 0 )
' Replace the path below before replaying the macro
var2 ( 0 ) = "E:\...\KwrTipCreateAssembly.CATPart"
Prod1.Products.AddComponentsFromFiles var2, "*"

Dim CstS1 As Collection
Set CstS1 = Prod1.Connections("CATIAConstraints")

Dim Ref1 As Reference
Set Ref1 = Prod1.CreateReferenceFromName(Plane.1)

Dim Ref2 As Reference
Set Ref2 = Prod1.CreateReferenceFromName(Pad.2 face)

Dim Cst2 As Constraint
Set Cst2 = CstS1.AddBiEltCst(1, Ref1, Ref2)
Cst2.Dimension.Value = 1.000000
Cst2.Orientation = 2

Dim Ref3 As Reference
Set Ref3 = Prod1.CreateReferenceFromName(Bottle axis)

Dim Ref4 As Reference
Set Ref4 = Prod1.CreateReferenceFromName(Cap axis)

Dim Cst3 As Constraint
Set Cst3 = CstS1.AddBiEltCst(2, Ref3, Ref4)

Prod1.Update
```

This macro can be started from a rule (see [About Rule Firing](#)) or directly by selecting the Tools->Macro->Run command from the standard menu bar.

About CATIA Automation Learning

There is a lot to be learned before you can write a complete macro in Visual Basic, but once you understand the basics of the language, you can be up and running in no time at all. Visual Basic is based on objects which have their own methods and properties. To set the value of a property, you follow the reference to an object with a period, the property name, an equal sign (=), and the new property value. At first sight, it is simple. The tedious thing when you have no previous programming skills is that the macro you record is a raw macro with objects and properties chained in one statement as in the extract below:

```
Dim ProductDocument0 As Document
Set ProductDocument0 = CATIA.Documents.Add ( "Product" )
Dim var1 ( 0 )
var1 ( 0 ) = "E:\...\KwrTipCreateAssembly.CATScript"
ProductDocument0.Product.Products.AddComponentsFromFiles var1, "*"

```

When you tackle automation, there are two ways to proceed:

- You already have a background in Visual Basic and you can refer to the list of *Programming Interfaces* provided with the CATIA documentation to write a macro or interpret a pre-recorded macro
- You are a beginner and just record a scenario. You must replay it to check whether the scenario has been actually recorded.

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.

This generic naming label appears when you record a .CATScript macro by using the Tools->Options->Macros command.

If you perform interactions such as selecting an edge or a face, the value specified for the arguments of some methods are written using generic naming. When you take a look at the macro, you can see arguments which are neither Visual Basic objects nor usual data. These arguments are written in a form similar to the simple example below:

Brp:(Pad.1:0(Brp:Sketch.1;1)).

You don't have to worry much about generic naming as the definitions relying on this technique are automatically inserted in CATIA macros.

Knowledgeware Automation

The objects below can be created and managed in a .CATScript macro:

- Parameters
- Formulas
- Design tables
- Rules and checks

Creating Knowledgeware Objects

The knowledgeware features are created from the collection which refers to their type. For example, to create a relation in a Part type feature, you must first retrieve the collection object containing the Part relations by using the Relations method on the Part object. To create a parameter in a Part, you must retrieve the collection object containing the Part parameters by using the Parameters method on the Part object.

Modifying Knowledgeware Objects

To manipulate a knowledgeware object, you just have to use the methods and properties of the relevant object.

An Example

Open the [KwrTipCreateAssembly.CATPart](#) document and run the KwrRelations.CATScript macro.

```
Sub CATMain()  
  
' Retrieve your active document - CATIA is your application  
' You get the active document by using the ActiveDocument property  
' on your application object  
Dim myDoc As Document  
Set myDoc = CATIA.ActiveDocument  
  
' Check whether the document is a CATPart  
' Analyse the pathname of the document  
' If the extension .CATPart is not found, a message is displayed  
' but you exit the procedure  
' InStrRev is a standard VB function  
Dim strPartName, strCATPart, myPos  
strPartName = myDoc.Name  
if (InStrRev(strPartName, ".CATPart", -1) = 0)_  
then MsgBox("Your document should be a .CATPart") : Exit Sub  
  
' Retrieve the collection object which contains  
' all the document relations  
' Activate all the relations  
' Display the relation names in a message box  
' Note: Statements below could not be applied to a CATProduct  
Dim strRel0 As String  
Dim strRel1 As String  
strRel1 = "Here is the list of relations" & vbCrLf & strRel0  
Dim myRelCol As Relations  
Set myRelCol = myDoc.Part.Relations
```

```
For Each myRel in myRelCol
myRel.Activate()
strRel1 = strRel1 & vbCrLf & myRel.Name
Next
Msgbox strRel1

' Make the configuration 4 active
Dim des1 As Relation
For Each myRel in myRelCol
  if myRel.Name = "DesignTable.1"_
  then Set des1 = myRelCol.Item("DesignTable.1"): des1.Configuration = 4
Next
CATIA.ActiveDocument.Part.Update
End Sub
```

This macro displays the list of relations within the document, changes the design table configuration and updates the document.

Tips

Operations that Cannot be Recorded

There are some operations that cannot be recorded but that can be programmed in a macro (the activation or deactivation of a relation for example). The list of objects, methods and properties you can actually use when writing a macro is given in the *Programming Interfaces*. Access to this documentation is provided on the CATIA documentation home page.

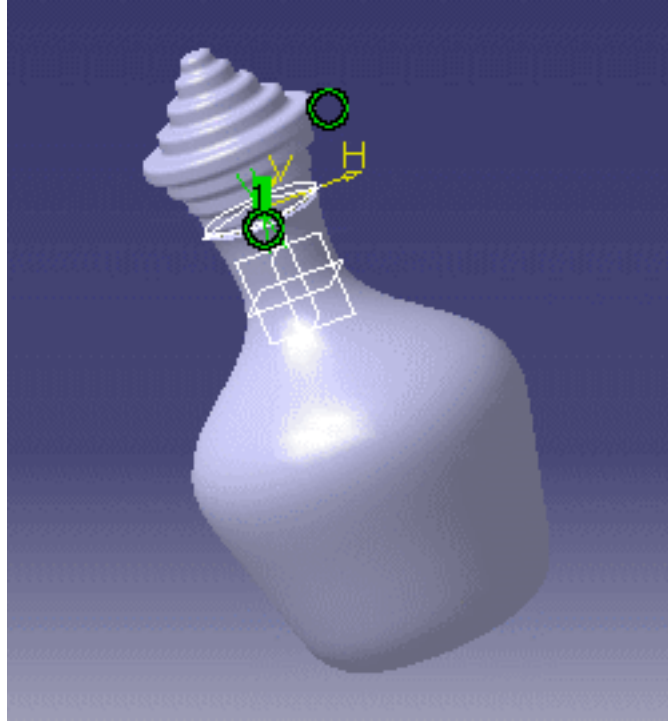
Retrieving Collections

Collections such as the Relations and Parameters objects can only be retrieved from a CATPart document. Prior to retrieving these collections, it is better to check the document type in your script, otherwise the macro crashes.

Retrieving a Collection Object

When retrieving a collection object by its name, it is better to check whether the object exists, otherwise the macro crashes.

Optimizing a Volume




Configuration 6
of [KwrProfilesDesignTable.xls](#)
applied to [KwrBottleProfiles.CATPart](#)

You now have a number of valid configurations. They are all stored in the form of a design table. Among the valid configurations, there is one that catches your attention. Configuration 2 is neat, but you would like to modify slightly its dimensions so that its internal volume is exactly 250 cm³. In the scenario below, we explain how to use the Product Engineering Optimizer to refine the results obtained for one of the bottles' shapes.

The Gradient Algorithm

1. Open the [KwrTipCreateAssembly.CATPart](#) document.

2. Access the Product Engineering Optimizer workbench and click the Optimize icon (). The Optimization dialog box is displayed.


3. Fill in the fields with the data below:

<i>Optimization type</i>	Target value
<i>Optimized parameter</i>	BottleVolume
<i>Target value</i>	250cm ³

<i>Free parameters</i>	xD initial value 82 mm lower bound: 0mm; upper bound: 200mm yD initial value 53mm lower bound: 0mm; upper bound: 200mm xE initial value 91 mm lower bound: 0mm; upper bound: 200mm yE initial value 26 mm lower bound: 0mm; upper bound: 200mm Don't specify any step.
<i>Algorithm</i>	gradient
<i>Termination criteria</i>	default values

4. Click the **Save Optimization data** box, otherwise you won't be able to save your optimization data.
5. Click **Run Optimization** to launch the algorithm. Don't intervene to stop the process. After the process has finished running, the target value is reached or almost reached. The values of the free parameters are displayed in the optimization box. Note that the results depend on the platform.

The Simulated Annealing Algorithm

1. Click **Cancel** in the Optimization dialog box.
2. Click the  icon or double click the optimization feature in the specification tree to display the optimization dialog box.
3. Fill in the fields with the values below:

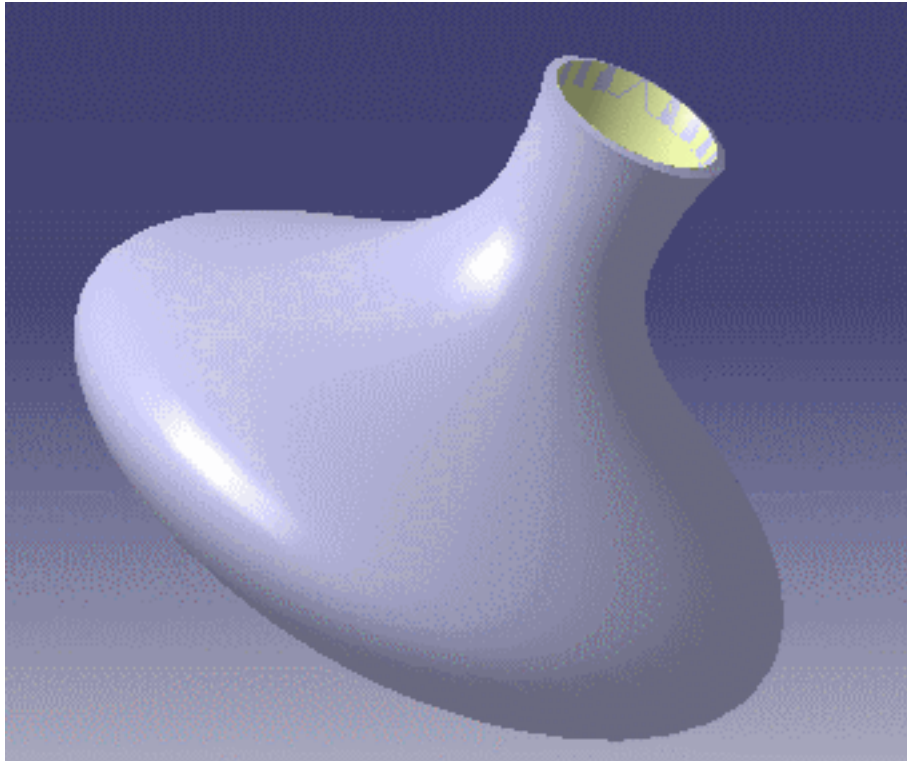
<i>Optimization type</i>	Target value
<i>Optimized parameter</i>	BottleVolume
<i>Target value</i>	250cm3
<i>Free parameters</i>	xD initial value 82 mm lower bound: 0mm; upper bound: 200mm yD initial value 53mm lower bound: 0mm; upper bound: 200mm xE initial value 91 mm lower bound: 0mm; upper bound: 200mm yE initial value 26 mm lower bound: 0mm; upper bound: 200mm <u>Specify a 0.1mm step for each free parameter.</u>

<i>Algorithm</i>	<u>Simulated Annealing - Fast</u>
<i>Termination criteria</i>	default values

4. Click the **Save Optimization** data box, otherwise you won't be able to save your optimization data.
5. Click **Run Optimization** to launch the algorithm. Don't intervene to stop the process.



Appendix

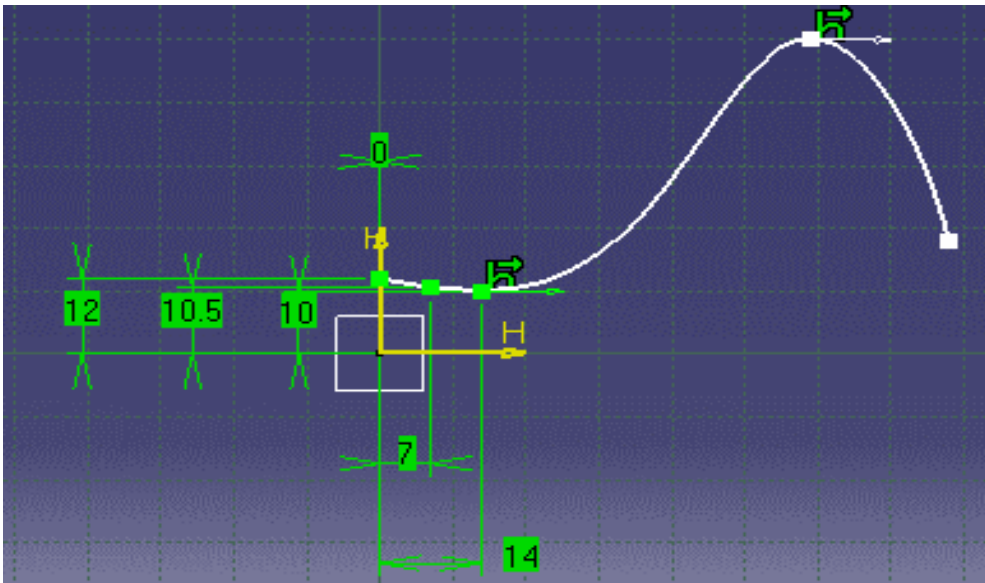
Creating a Deformable Revolution Body



The bottle to be designed will be a five point spline rotated around an axis, and thickened. Once it is created, the spline is rotated, the open end corresponding to the bottom of the bottle is closed and the resulting shape is thickened.

Creating the Initial Spline

1. Create a new part.
2. In the Part Design workbench, access the Sketcher by clicking the  icon.
3. Click the  icon to draw a five point spline looking something like the curve below.



4. Specify the constraints given in the table below.

	Location on spline	Offset along H	Offset along V	Tangency constraint
Control point A	Extremity on the neck	$x_A = 0\text{mm}$	$y_A = 12\text{ mm}$	none specified
Control point B	Next to A on the neck	$x_B = 7\text{mm}$	$y_B = 10.5\text{mm}$	none specified
Control point C	Next to B on the neck	$x_C = 14\text{mm}$	$y_C = 10\text{ mm}$	horizontal
Control point D	Next to C	none specified	none specified	horizontal
Control point E	Other extremity	none specified	none specified	none specified



Use the icon to specify offset constraints on the three control points making up the bottle neck.



Use the icon to specify an horizontal tangency on points C and D.


At this stage, don't specify any offset on point D and E otherwise you won't be able to modify the bottle profile. But if you wish to start from a spline similar to the one below, you can set the D coordinates to 59 mm (along H) and 50 mm (along V) and the E coordinates to 77 mm and 17mm.

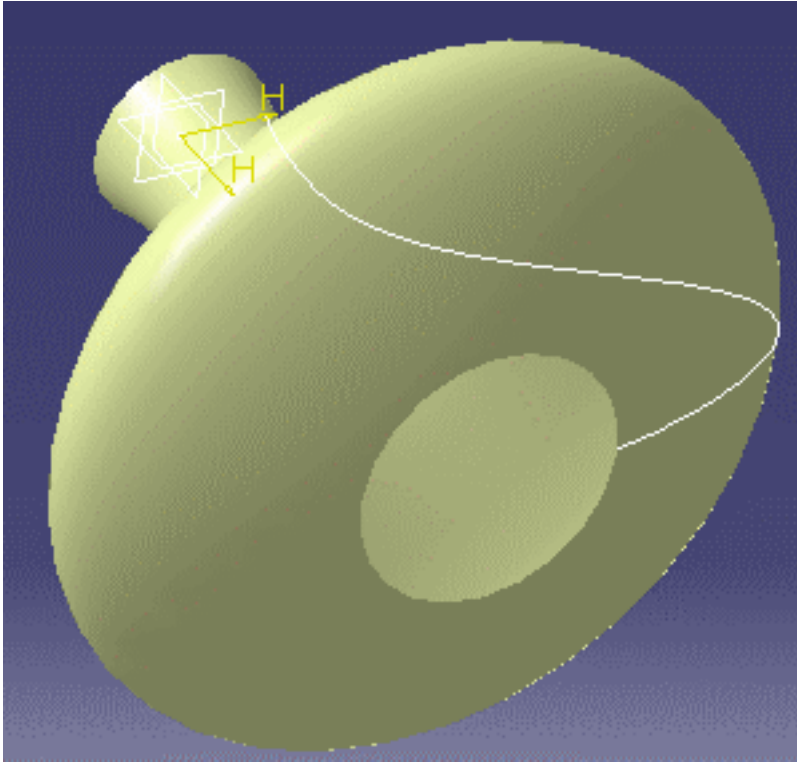



5. Click the icon to access the formulas dialog box, then modify the offset names according to the names given in the table above.

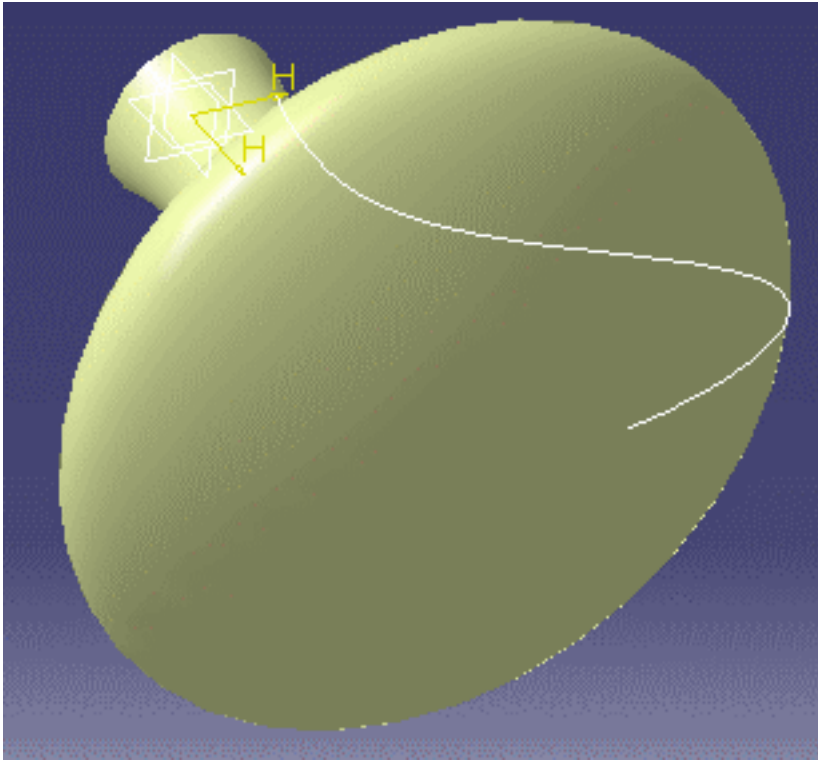
The resulting document is KwrInitialSketch.CATPart

Creating the Revolution Surface

1. Access the Generative Shape Design workbench and use the  icon to rotate the spline created above around the H axis. Do a complete rotation (0deg - 360deg). The generated surface is opened at both ends. This is what your revolution surface looks like from the bottom.




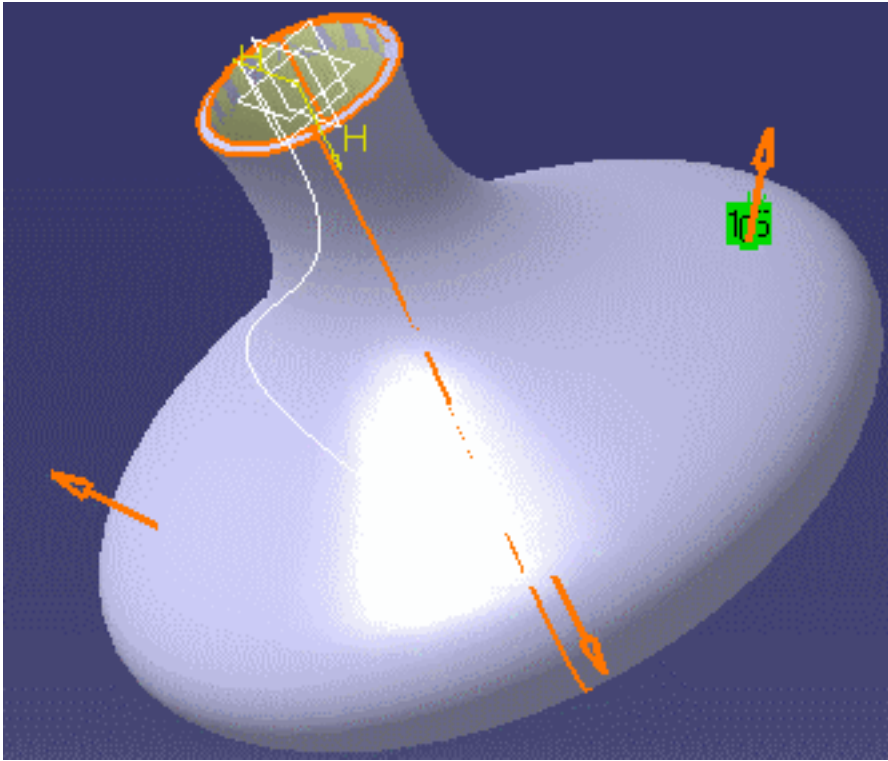
2. Access the Generative Shape Design workbench and use the  icon to close the bottle lower part. This is now what you should get onscreen.



3. Use the  icon to join the Revolute and Fill type features.

Thickening the Revolution Surface

1. Access the Part Design workbench and use the  icon to thicken the joined surface. Specify a 1.5 mm as First Offset and 0 mm as Second Offset. This thickness should be an external thickness. The arrows displayed on screen when you apply the offset values should be directed toward the outside of the bottle (see the figure below).



2. Save the resulting document.

The resulting document is KwrThickSurface.CATPart

When applying a thickness to the joined surface, we could specify a first offset as well as a second offset. But this would make things a bit more complicated as later on we want to measure the internal volume of the bottle. This internal volume is to be calculated from the "Volume" measure of a surface. Adding an internal thickness entails subtracting the volume resulting from this internal thickness to the one calculated for the joined surface.

Now you can select the sketch feature in the specification tree to access the sketcher, drag the D and E points and see how the global shape reacts.

CATIA Knowledgeware Infrastructure

[Relations](#)

[Parameters](#)

[Formulas](#)

[Design Tables](#)

Tips and Techniques - Summary

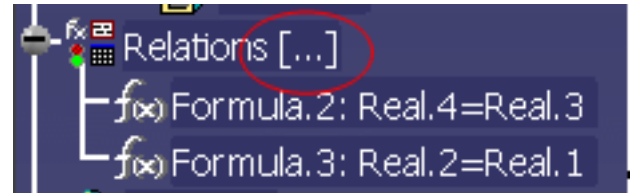
This part is intended for those of you who need a quick answer to their questions about the knowledgeware capabilities. However, using this part requires a prerequisite knowledge of the *CATIA* basic concepts.

Relations

Hiding relations

You can hide a knowledge relation (formula, rule, check, ..) by right-clicking this relation in the specification tree and by selecting the **Hide** command.

- A visual indicator located at the relations set level indicates that the set contains hidden relations. Note that this indicator is not recursive.
- If the user tries to delete a relations set containing hidden relations, a message displays asking the user if he wants to delete the relations set that contains hidden relations. Note that if the user tries to delete a relations set containing another relations set with hidden relations, the message will display.



Parameters

[Switching between Simple and Multiple Values After Creating a Parameter](#)

[Creating a parameter](#)

[Creating a multiple-value parameter](#)

[Deleting a parameter](#)

[Displaying parameters in the \$f\(x\)\$ dialog box](#)

[Displaying parameter values in the geometry area](#)

[Updating parameters values](#)

[Declaring a parameter as constant](#)

[Editing or modifying a parameter](#)

[Editing constrained parameters](#)

[Adding a comment to a parameter](#)

[Associating a URL with a parameter](#)

Hiding a parameter from the specification tree

Importing parameters

Ordering Parameters

Specifying a parameter value as a measure

Specifying a tolerance


Specifying lower and upper bounds

Specifying an increment/decrement amount

Specifying the Material parameter value

Valuating a Boolean with a Predicate

Creating a Parameter

Click . In the Formulas Editor, select a type in the **New Parameter of type** list, then click **New Parameter of type**. The new parameter is added to the parameters list. A default name is given to the parameter. If need be, rename this parameter.


Creating a multiple-value parameter

To assign multiple values to a parameter, select the **Multiple values** option at parameter creation and enter the values one by one to specify the value list.

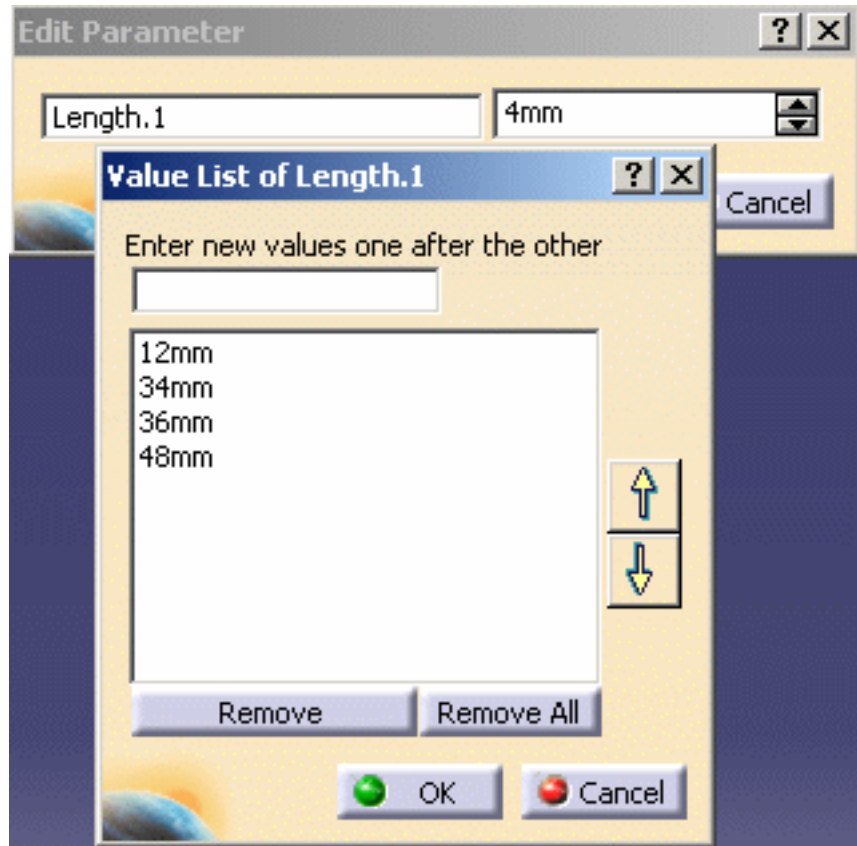
Switching between Simple and Multiple Values After Creating a Parameter

It is possible to add values to a single value parameter after creating it. To do so, use one of the 2 methods described below:



First Method

1. Click the  icon in the Knowledge tool bar. In the Formulas Editor, select a type in the **New Parameter of type** list, then click **New Parameter of type** with **Single value**. The new parameter is added to the parameter list. A default name is given to the parameter. If need be, rename this parameter.
2. Click **OK** when done.

3. Double-click the parameter in the specification tree. The **Edit Parameter** window displays.
4. Right-click the value field of the **Edit Parameter** window and select the **Add Multiple Values ...** command.
5. In the **Value List** window, enter the new values.
6. Click **OK** when done.

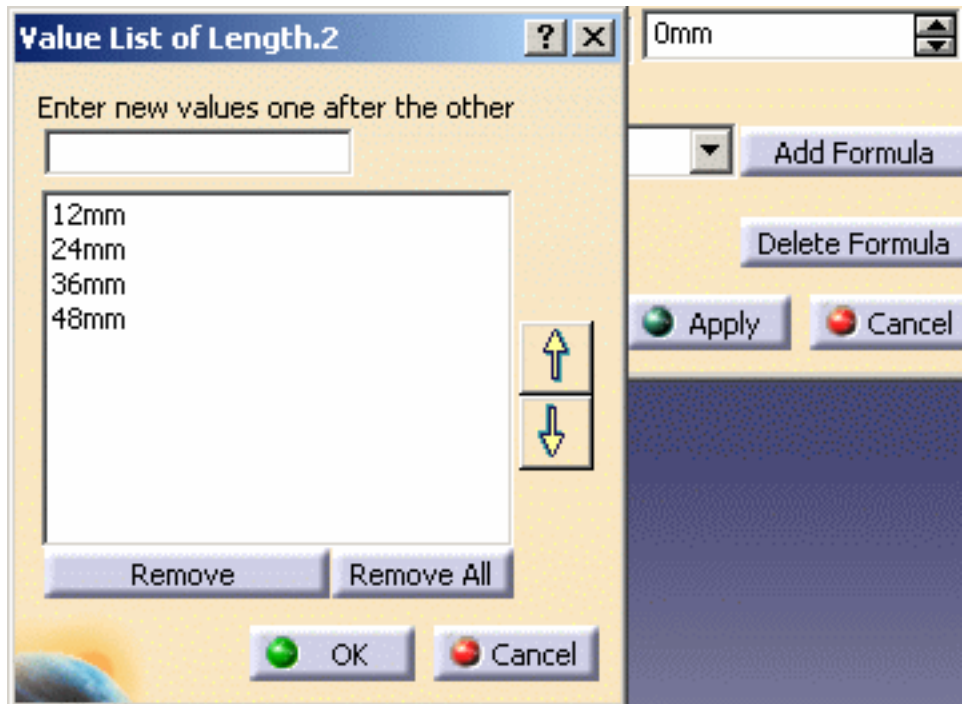


Second Method

1. Click the  icon in the Knowledge tool bar. In the Formulas Editor, select a type in the **New Parameter of type** list, then click **New Parameter of type** with **Single value**. The new parameter is added to the parameter list. A default name is given to the parameter. If need be, rename this parameter.
2. Click **OK** when done.
3. Click the  icon in the Knowledge tool bar. In the Formulas Editor, select the parameter that you have just created.
4. Right-click the Value field of the **Edit name or value of the current parameter**


field and select the **Add Multiple Values ...** command.

5. In the **Value List** window, enter the new values.
6. Click **OK** when done.



Deleting a parameter

There are two ways to proceed:

- Click the  icon in the Knowledge tool bar. In the Formulas Editor, select the parameter to be deleted, then click Delete Parameter.

Or

- In the specification tree, right-click the parameter to be deleted, then select the Delete command from the contextual menu.

Only user parameters can be deleted (the Material parameter cannot be deleted).

Displaying parameters in the $f(x)$ dialog box

The parameter list displayed in the Formulas Editor depends on the selected feature. If you select the document root feature in the specification tree, you display all the document parameters. If you select a given feature in the specification tree, you display only the parameters related to this feature.

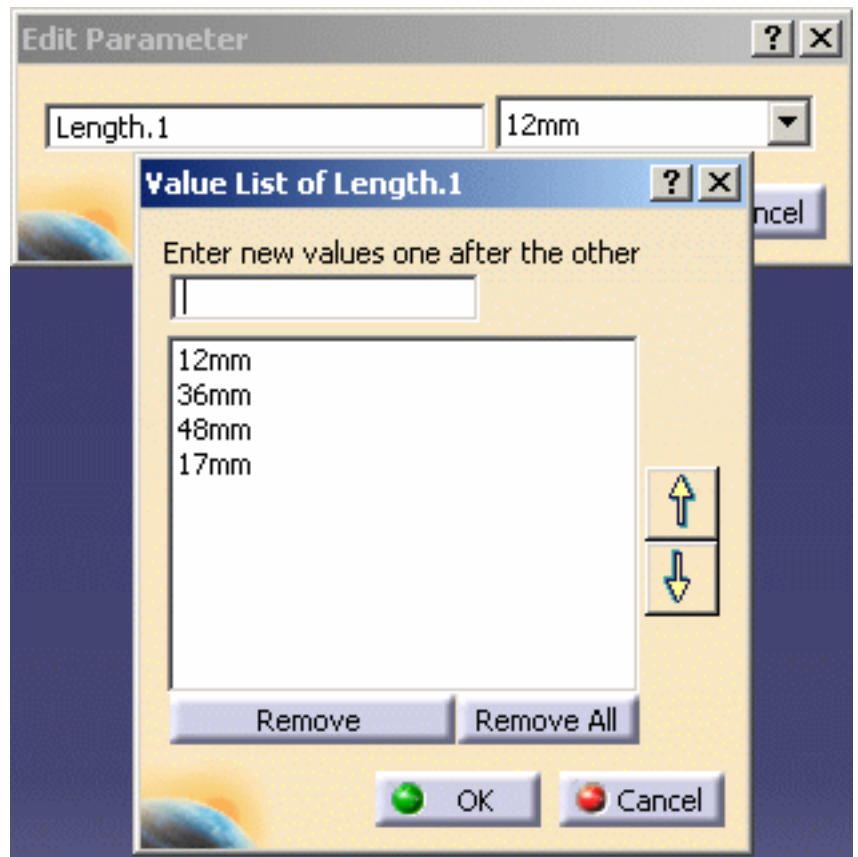
Displaying parameter values in the geometry area

To display parameter values in the geometry area, you must have the Formulas Editor open. Selecting a parameter in the parameter list will highlight this parameter in the specification tree and display its value in the geometry area.

Updating parameters values

To update the values of a multiple values parameter, proceed as follows:

1. Double-click the parameter in the specification tree. The Edit Parameter window displays.
2. Right-click the value field of the window, and select the Update Values... command. The Value List window displays.
3. Edit the values and click OK when done.




Declaring a parameter as constant

Select the Properties command from the parameter contextual menu. In the Parameter properties tab, check the Constant option.

Editing or modifying a parameter

There are three ways:

- Click . Select the parameter to be edited in the Formulas Editor, then modify its value in the **Edit name or value of the current parameter** field.
- In the specification tree, double-click the parameter to be edited, then modify its value in the **Edit Parameter** editor.
- In the specification tree, right-click the parameter to be edited, then select the **Parameter object->Definition...** command from the contextual menu.

Editing constrained parameters

When a parameter is constrained, a push button is provided opposite the value field of its edition box. This push button represents the relation which constrains the parameters. Clicking this button displays the editor of the relation which constrains the parameter (the formula editor or the rule editor for example).

Adding a comment to a parameter

There are two ways to proceed:

- Edit the parameter, then in the value field of the parameter editor, select the **Edit Comment...** command from the contextual menu.
- Select the Properties command from the parameter contextual menu. In the **Parameter** properties tab, fill in the Comment field.

Associating a URL with a parameter


Access the Knowledge Advisor workbench, then click the **Comment & URLs** icon.

Hiding a parameter from the specification tree

There are two ways to proceed:

- Edit the parameter, then in the value field of the parameter editor, select the **Hide** command from the contextual menu.
- Select the **Properties** command from the parameter contextual menu. In the **Parameter** properties tab, check the **Hidden** option.
- Right-click the parameter and select the **xxx object->Hide** command.

Importing parameters

Click the  icon in the Knowledge tool bar. Then click **Import...** A file selection dialog box is displayed. Select either a .xls file (Windows only) or a .txt file. If the imported parameters already exist in the document, the import process automatically updates the document.

Ordering Parameters

To reorder parameters, right-click the parameter to be reordered, and select the **Reorder...** command. The **Reorder** dialog box displays. In the specification tree, select a parameter below which the parameter to be reordered will be located. Click **OK** when done.

Specifying a parameter value as a measure

Edit the parameter, then in the value field of the parameter editor, select the **Measure Between** command from the contextual menu. The **Measure Between** dialog box is displayed. You can use this dialog box to measure the distance between two edges for example. Select **Between as measure** type, then select successively two edges in the geometry area. The distance between both edges is displayed in the Results pane.

Specifying a tolerance (Length or Angle parameter only)

The default tolerance specified in Parameters Tolerance tab of the **Tools->Options->General->Parameters and Measure** settings can be redefined. To do this, edit the parameter, then in the value field of the parameter editor, select the **Tolerance->Edit...** command from the contextual menu.

Specifying lower and upper bounds

Edit the parameter, then in the edition box, right-click the value field(s) and select the **Add Range...** command from the contextual menu. This capability does not apply to parameters such as booleans or strings.

Specifying an increment/decrement amount


Edit the parameter, then in the edition box, right-click the value field(s) and select the **Change step...** command from the contextual menu. Select **1mm** or click the **New one** command. In this case, the New step dialog box displays and you can enter another step.

Note that:

- This capability does not apply to parameters such as booleans or strings.
- This information is not stored and will be lost when closing the *CATIA* session.

Specifying the Material parameter value

There are two ways to proceed:

- Select the part you want to apply the material to, then click the  icon. Select one of the material in the library which is displayed. Prior to doing this, check that the material catalog is installed on your machine.
- Edit the material parameter from the specification tree, then enter the new material in the parameter editor.

Valuating a boolean with a predicate

It is now possible to valuate a boolean with a predicate instead of using operators. See the example provided in the [KwrPredicate.CATPart](#) file.

Formulas

[Creating a formula](#)

[Quickly Accessing Parameters in the Formulas Editor](#)

[Using the dimensions of a sketch in a formula](#)

[Editing or modifying a formula](#)

Displaying only a formula definition in $f(x)$

Specifying a measure in a formula

Activating / deactivating a formula

Importing parameters and formulas



Deleting a formula



Please note that in the Formulas editor, the default step is 1 whatever the unit.

Creating a formula

There are three ways to proceed:


- Click the  icon in the Knowledge tool bar. Select the parameter to be constrained, then click **Add Formula**. Enter the formula in the Formulas Editor.
- Click the  icon in the Knowledge tool bar. Double-click the parameter to be constrained and enter the formula in the Formulas Editor.
- Edit a parameter, right-click the value field(s), then select **Edit formula...** from the contextual menu.

Quickly Accessing Parameters Values in the Formulas Editor


A new field enables the user to change a parameter value when creating and especially editing a formula.



This field is especially useful when you want to modify the value of a parameter valued by a formula. To do so, proceed as follows:

- Double-click the parameter in the specification tree.
- Click the  icon located next to the parameter value field. The Formula Editor opens.
- In the Relation body, select the parameter to be modified and change its value. To get an example, see below.


1. Open the [KwrPredicate.CATPart](#) file.


2. Click the  icon in the Knowledge tool bar.

3. Create 3 parameters of length type with the following values:

- Length.1= 12mm
 - Length.2= 14mm
 - Length.3= 0mm
4. Select the Length.3 parameter in the Parameter list and click the **Add Formula** button. The Formula Editor displays.
 5. Enter the following formula into the editor and click **OK** when done:

Length.3= Length.2
*Length.1

6. Double-click Length.3 in the specification tree. The Edit Parameter dialog box displays.
7. Click the  icon located next to the parameter value field. The Formula Editor displays.
8. In the Editor, select Length.2 in the Members of All column and set its value to 9mm in its value field.

Length.2	9mm 
----------	------------------------------------------------------------------------------------------

9. Click **OK** when done. The Length.2 and Length.3 values are updated in the specification tree.

Using the dimensions of a sketch in a formula

Unless sketch dimensions are declared as constraints, you cannot manipulate them as parameters in relations. This applies for example to the radius of a circular sketch or to the width and length of a rectangular sketch.

Editing or modifying a formula

There are four ways to proceed:

- In the specification tree, right-click the formula to be edited, then select the **Formula object->Definition** command from the contextual menu.
- In the specification tree, double-click the formula to be edited.
- In the specification tree, double-click the user parameter, then click the **f(x)** button in the **Edit Parameter** dialog box.
- Edit a parameter, right-click the value field(s), then select **Formula->Edit** from the contextual menu.

Displaying only a formula definition in $f(x)$


When you select a formula in the specification tree, then click the Formulas icon, the parameter list displays the constrained parameter, the formula activity and the parameters used as input in the formula.

Specifying a measure in a formula


- Edit the parameter, right-click the value field in the parameter edition box, then select the **Edit formula...** command from the contextual menu.
- In the formula editor, click the **Wizard** button, select the **Measures** item in the dictionary then select one of the measure functions displayed in the wizard.
- Fulfill the function prototype and allow for the required number of arguments. To enter an argument value, position the cursor where the argument is intended to be and capture the feature definition from the specification tree or from the geometry area.

Activating / deactivating a formula

There are three ways to proceed:


- Click the  icon in the Knowledge tool bar. In the parameter list of the **Formulas** Editor, select the formula/activity parameter and modify its value in **Edit name or value of the current parameter**.
- In the specification tree, right-click the formula whose activity is to be modified, then select the **Formula object->(De)Activate** command from the contextual menu.
- Edit the constrained parameter, right-click the value field(s), then select the **Formula->(De)Activate** command from the contextual menu.

Importing parameters and formulas

Click the  icon in the Knowledge tool bar. In the $f(x)$ dialog box, click Import, then specify the import file path. The import file should be either a .txt file or a .xls file. The parameters and formulas are applied to the document. If the imported parameters already exist in the document, the document is automatically updated by the import process.

Deleting a formula

There are three ways to delete a formula:

1. Click . In the parameter list of the $f(x)$ dialog box, select the parameter which is constrained by the formula to be deleted, then click **Delete Formula**.

2. In the specification tree, right-click the formula to be deleted, then select the **Delete** command from the contextual menu.
3. Editing the parameter. In the parameter editor, right-click the value field(s), then select the **Formula->Delete** command from the contextual menu.

Design Tables

[Creating a design table from the document parameter values](#)

[Creating a design table from a pre-existing file](#)

[Editing a design table](#)

[Adding a column to a design table](#)

[Adding Ranges and Values to a Design Table](#)

[Making a query in a Design Table](#)

[Rearranging columns in a Design Table](#)

[Selecting the Design Table Editor on Unix](#)

[About the parameters that cannot be inserted in a design table](#)

[Tips about selecting the parameters to be inserted in a design table](#)


[Tips about using the access functions to the design table](#)

[About the broken status](#)

[Deleting a design table](#)


Creating a Design Table from the Document Parameter Values



Click the  icon in the Knowledge tool bar. Check the option Create a design table with current parameter values. Select the parameters to insert, then specify the .xls (Windows) or .txt file where the design table is to be created.

Creating a Design Table from a pre-existing File



Click the  icon. Check the option Create a design table from a pre-existing file. Specify the file containing the design table data, then create the necessary associations.

Editing a Design Table

There are two ways to proceed:

- In the specification tree, double-click the design table to be edited. The *CATIA* design table is displayed. The active configuration is highlighted.

Or

- In the specification tree, right-click the design table to be edited, then select the DesignTable.object->Edition command from the contextual menu.

Adding a Column to a Design Table

To add a column to a design table, open the .txt file in a text editor or the .xls file in Excel, and add a column to the file.

Adding Ranges and Values to a Design Table

It is possible for the user to add ranges and default values to a design table. The ranges will appear between [;], and the default values between <> (see example below).

Length.1 (mm)[-20;20]	Length.2 (mm)	Length.3 (mm)[;]
[-5;5]<2>	0	0
10 <5>		[2;]

It is possible to specify a range:

- At the top of a column: in this case the range will apply to all cells if no other range is specified in the cells. In the table above, the range that applies to cell A2 is [-5;5] and the one that applies to cell A3 is [-20;20].
- In a specific cell: it is possible to add a range to a specific cell (see cell A2 in the table above). In this case, the range specified in this cell will apply to this cell and to the cells located below if no range is specified at the top of the column.
If a range is specified at the top of the column, the range indicated in the cell applies only to this cell. In the table above, [-5;5] applies only to cell A2 because a range [-20;20] is specified at the top of the column. If no bounds were indicated at the top of the column, the range indicated in cell A2 would also apply to cell A3.

Note that:

- When specifying a range, only one of the bounds may be indicated. ([2;] in the table above for example).
- Ranges contained in the Design Table override those you might have added to your parameters by using the **Add Ranges...** command.
- The default value specified between <> (see table above, column 2) indicates that the parameter is not constrained by the Design Table.
- If a range is specified but no default value, the parameter value is not constrained by the Design Table.

Deleting a Design Table

In the specification tree, right-click the design table to be deleted, then select the Delete command from the contextual menu.

Making a Query in a Design Table

The syntax which is authorized in the Filter field of the Design Table dialog box (Configurations tab) is the same as for Knowledge Advisor checks. You can type the filtering relation directly in the appropriate field or click the Edit... button to access the language dictionary.

An example of a query relation: `TangE > 10 deg and yE > 10mm`

Rearranging Columns in a Design Table

The way columns are arranged in a design table is identical to the way the associations were defined. You can now rearrange the columns without deassociating the parameters and the columns and associating them again just afterwards. To do so, click the **Associations** tab in the Design Table window, and use the **Up** and **Down** arrow buttons.

Selecting the Design Table Editor on Unix

The environment variable `CATTermDT` enables the user to choose the terminal from which the design tables editor will be started. The editor can be set up by using the environment variable `CATTextEditorDT`. `xterm` is used by default.

This way, the user can select the terminal, which can impact the way the editor behaves. For example, for Japanese users, if `vi` is launched from `xterm`, `vi` will not support kanji whereas if it is launched from `aixterm` (AIX), kanji will be supported.

Example: `export CATTermDT=aixterm`

About the parameters that cannot be inserted in a design table

Only parameters which are not already constrained by any other relation or by any other design table can be used to create a design table. If a parameter is already constrained, it does not appear in the Parameters to insert list in the design table dialog box.

Tips about selecting the parameters to be inserted in a design table

The Filter Name and Filter Type filters can be used to restrict the display of a parameter list. If you specify `x` in the Filter Name field of the Select parameters to insert dialog box, you will display all the parameters with the letter `x` in their name (`xA`, `xB`, `xC`, `xD`, `xE`). If you select the Renamed Parameters in the Filter Type list, you will display all the parameters you have renamed in the Formulas Editor (`yA`, `xB`, `xA`, `yC`, `xC`, `yB`, `yD`, `xD`, `yE`, `xE`, `TangE`).

Parameters to be inserted can be multi-selected. You just have to keep on pressing the `Ctrl` key while you select parameters. If you do this, the group of multi-selected parameters will be carried forward onto the Inserted parameters list in the order in which they are displayed in the initial list.

When the design table is created, the rank of the columns fits the rank of the parameters in the Inserted parameters list. If you want to have columns ordered in a given way in the design table, you must insert the parameters one by one.

Tips about the access functions to the design table

Once in the formula (rule or check) editor, select the Design Table item in the dictionary, the list of the methods that can be applied to a design table is displayed. Select a method, then click `F1` to display the associated documentation.


About the broken status

When the active configuration does not fit the set of configurations resulting from a query operation, you get a message prompting you to deactivate the filter (uncheck the filter box) or change your configuration.

If neither action is undertaken, your design table takes on a broken status. In the specification tree, a broken or invalid design table is displayed.



Optimal CATIA PLM Usability



When working with ENOVIA V5, the safe save mode ensures that you only create data in CATIA that can be correctly saved in ENOVIA.

ENOVIA V5 offers two different storage modes: Work package (Document kept - Publications Exposed) and Explode (Document not kept).



Please find below the list of the Knowledgeware commands along with their accessibility status in Enovia V5.

Note that the restrictions listed below apply only when working at the Product level. They do not apply when working in a Part context.

Commands	Accessibility in Enovia V5 (Explode Mode)	Comments
Formula Editor	Available in limited mode	The user can only edit existing formulas and parameters. He cannot create new ones.
Equivalent Dimensions	Not available	None
Design Table	Available. See Working with Design Tables in ENOVIA LCA .	None
Law	Not available	None
Knowledge Inspector	Available	None
Check analysis toolbox	Available	None
Lock/Unlock	Available	None
Comment & URLs...	Available	None




Using the Feature Dictionary Editor




- Starting the Feature Dictionary Editor
- Creating a New Object Class
- Adding Properties to an Object Class
- Defining Discrete Values for a Property
- Generating a Report
- Creating a New Feature Dictionary
- Opening a Reference Dictionary
- Modifying the Object Naming Rules


Starting the Feature Dictionary Editor



 This task shows you how to start the Feature Dictionary Editor.

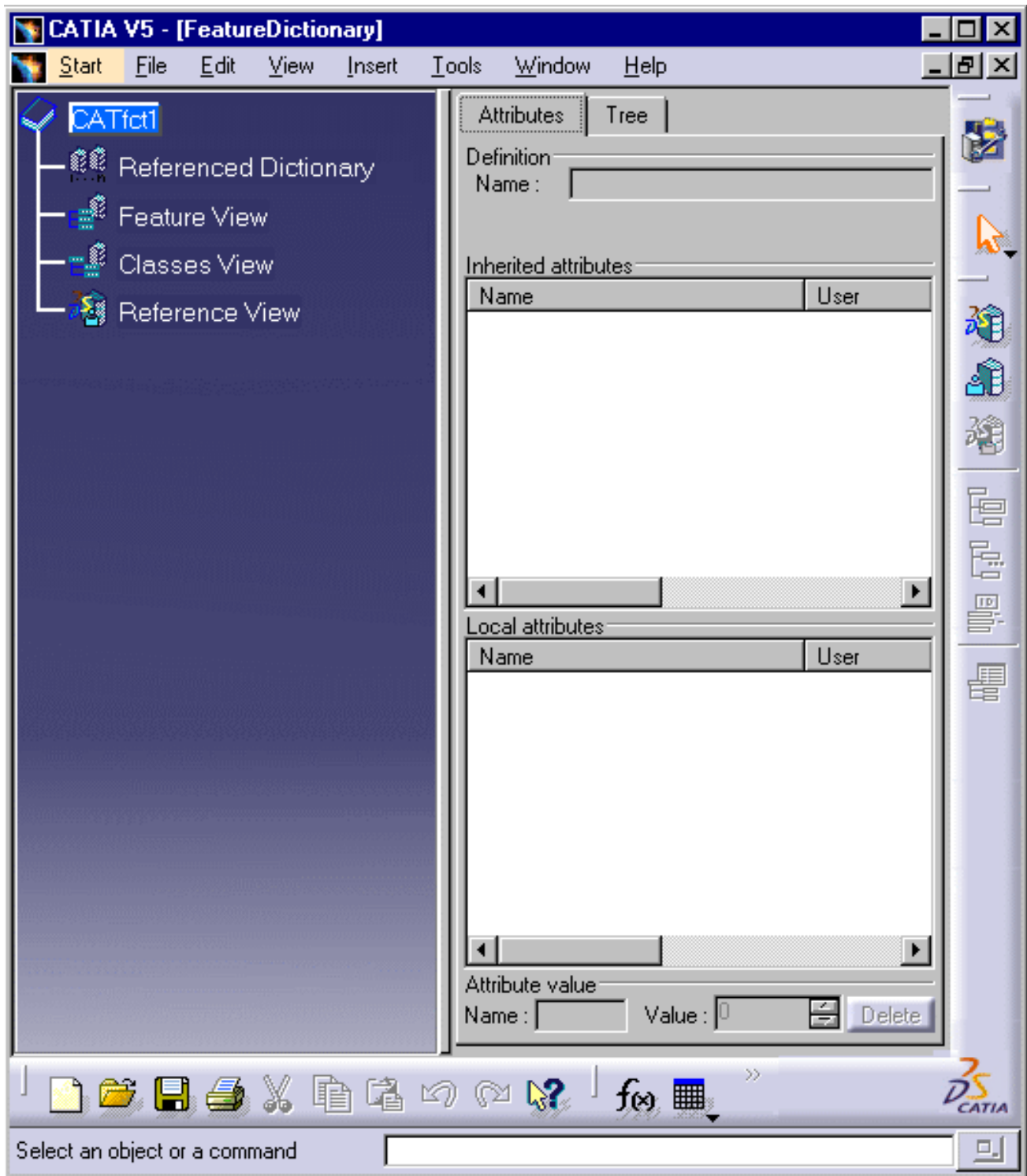
 The feature dictionary editor allows you to create delete and manage object classes. Object classes are classifications under which you create various objects, like components, for storing in the catalog. You may, for instance, want to have several object classes under valve_function, one of them being check_valve_function, and create various types of check valve under the class.

 **1. Click Start -> Infrastructure -> Feature Dictionary Editor.**

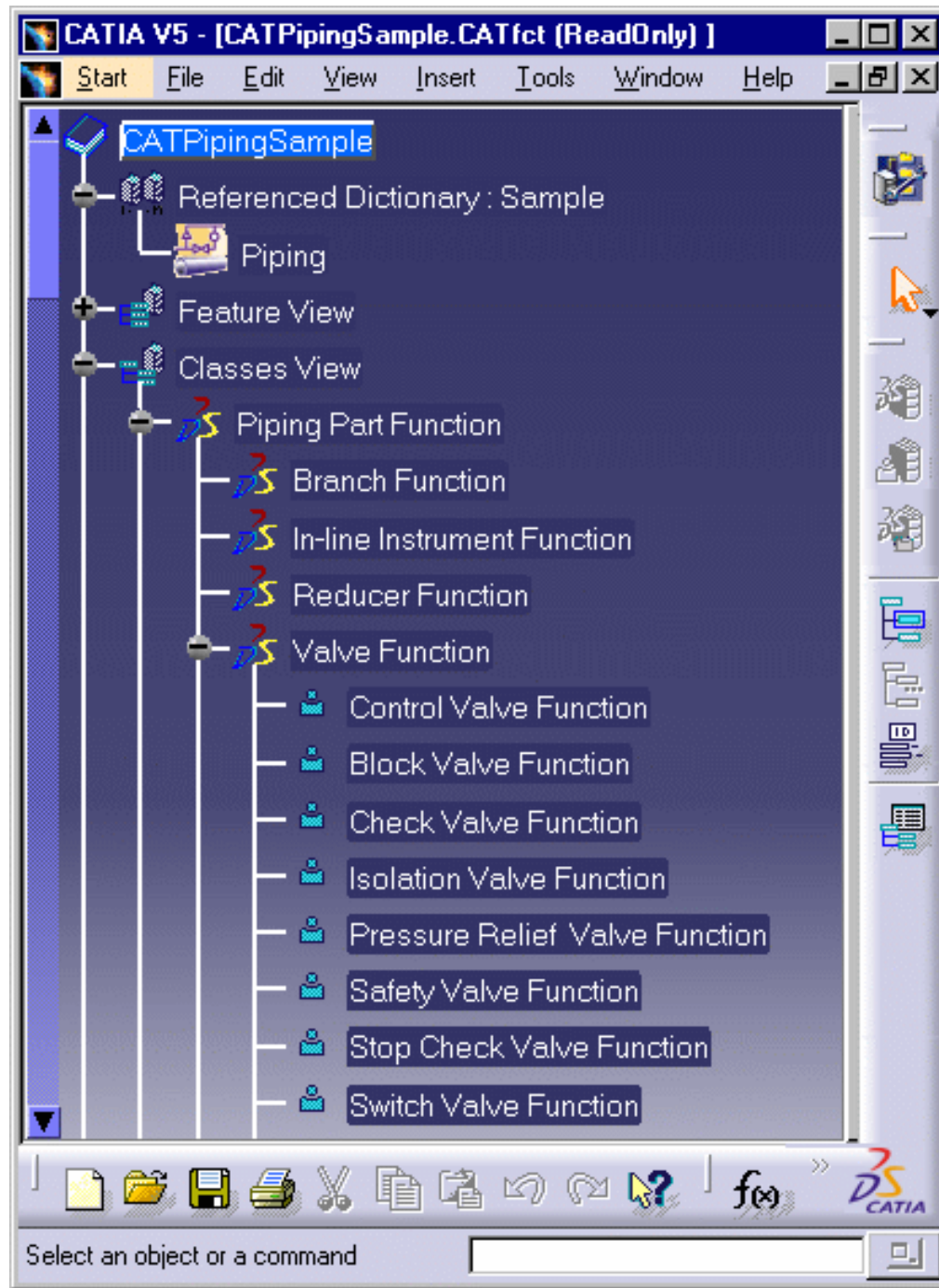
The Feature Dictionary Editor  opens.

The specifications tree displays three views. Referenced Dictionary will display under it any external dictionary files that are referenced in the document. Feature View will have under it all classes defined in the document. Classes View will display all classes available in the document, including the predefined classes that are included with the application.

On the right side, Inherited View shows attributes that a class inherited from its super class. Local Attributes shows attributes added specifically to a class.



2. To view the feature dictionary for a specific application (such as Piping Design) you need to open the .CATfct file associated with it. The .CATfct file contains all the classes. To open it click **File - Open** and navigate to **intel_a\resources\graphic** and open the relevant .CATfct file, such as **CATPipingSample.CATfct** for Piping Design. The file will open in the feature editor and you will be able to see all existing classes, and make changes if you need to.



Creating a New Object Class


P2



This task shows you how to create a new object class.

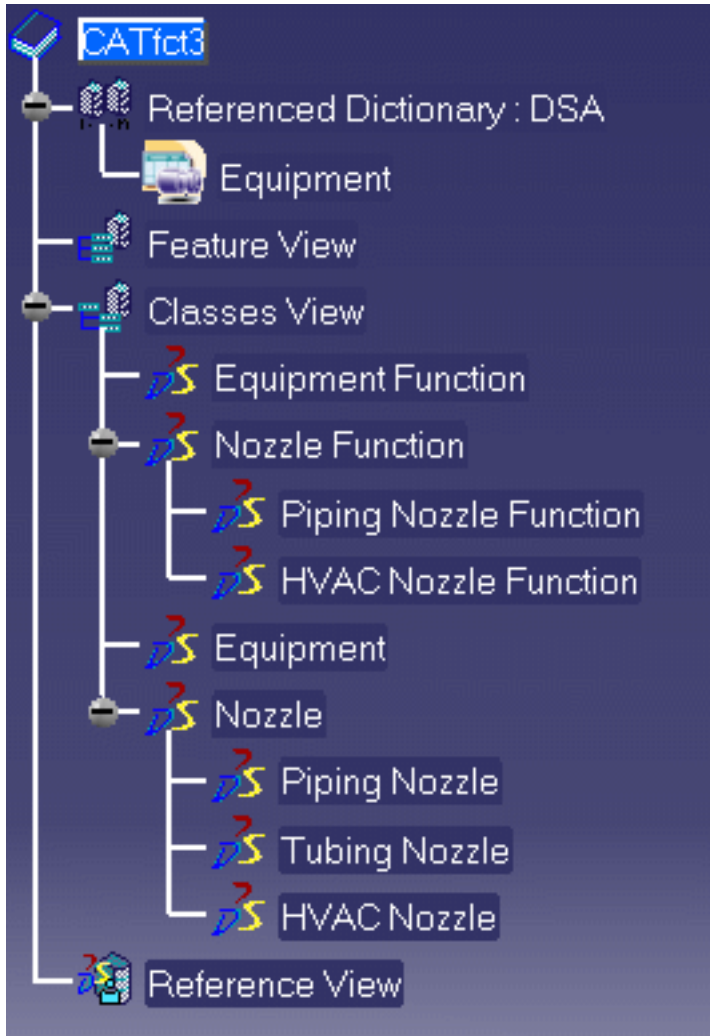


1. From **Start**, go to **Infrastructure** and click on **Feature Dictionary Editor**. Click the

Open Application Dictionary button . The Open Application Dictionary dialog displays.



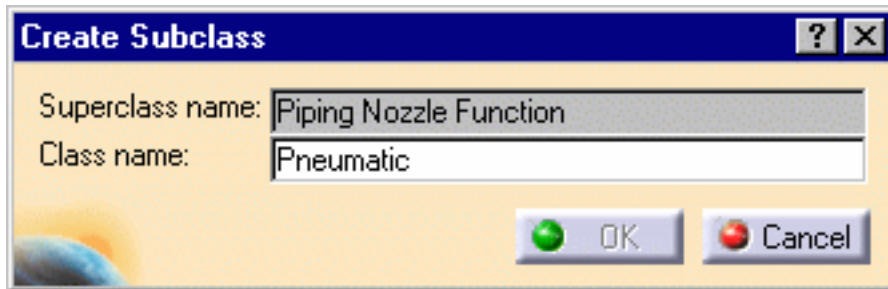
2. Click on the down arrow and select one of the categories. Enter a name in the **Client ID** field. This will appear next to the classes in the specifications tree. Click **OK**.
3. The classes that are available to the document display in the specifications tree under Classes View. The object classes that will be displayed are the base classes included with the application. You cannot rename or delete them, but you can create object classes under them.



4. Double-click the object class under which you want to create the new class, then click the

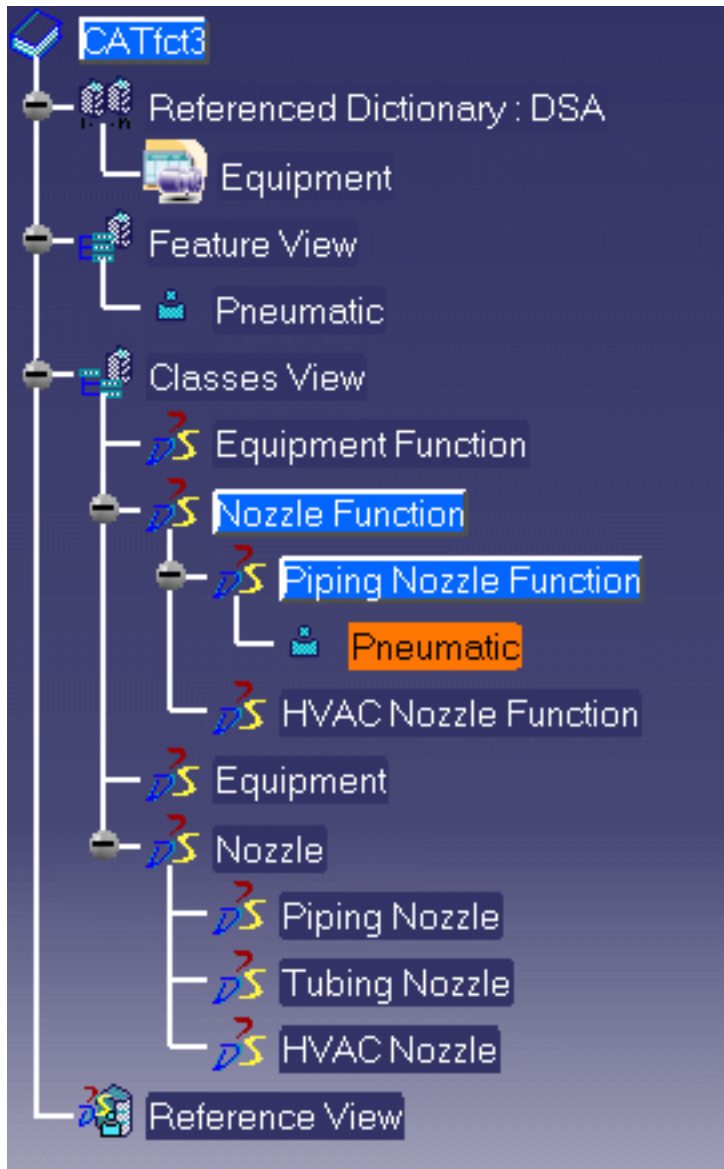


Create Subclass button. The Create Subclass dialog box displays.



5. Enter a name in the **Class name** field and click **OK**.

- 6. The new object class is created and displays in the specifications tree under Feature View and Classes View.



Adding Properties to an Object Class

P2



This task shows you how to add properties to an object class.



1. In the specifications tree, select the class to which you want to add a property. When you select a class its existing properties show under the Inherited Attributes and Local Attributes windows. Inherited attributes are those inherited from the super class to which this class belongs. Local attributes are properties added to the class itself.



2. Click on the **Add Attributes** button. The Add Attribute dialog box will display.

Add Attribute [?] [X]

Selected class name : Equipment

Attribute Name : []

Attribute type : Real

With : Single Value

Default Value : 0 [] [Unset]

[OK] [Cancel]

3. Enter an attribute name.

Click on the down arrow in the **Attribute type** field and select an attribute type.

Click on the down arrow in the With field and select **Single Value** or **Discrete Values**. If you select Single Value, you will be able to change the value later by using the Edit - Properties command and entering a new value. If you choose Discrete Values, you will only be able to select a value from a predefined list. See [Defining Discrete Values for a Property](#) to find out how to create and save this list of values.

Enter a default value.

4. Click OK.

The new attribute will display in the Local Attributes window.



Defining Discrete Values for a Property



When you [add properties to an object class](#), you have to select whether you want a single value or discrete values. If you select discrete values, it means that the property will have predefined values - the user will not be able to enter a value but will have to select from a predefined list. Those values have to be created in a text file and stored in a specific directory.

You also need to be familiar with project resource management to be able to create and store those discrete values.

Please refer to [Understanding Project Resource Management](#) in this guide.



Generating a Report

P2



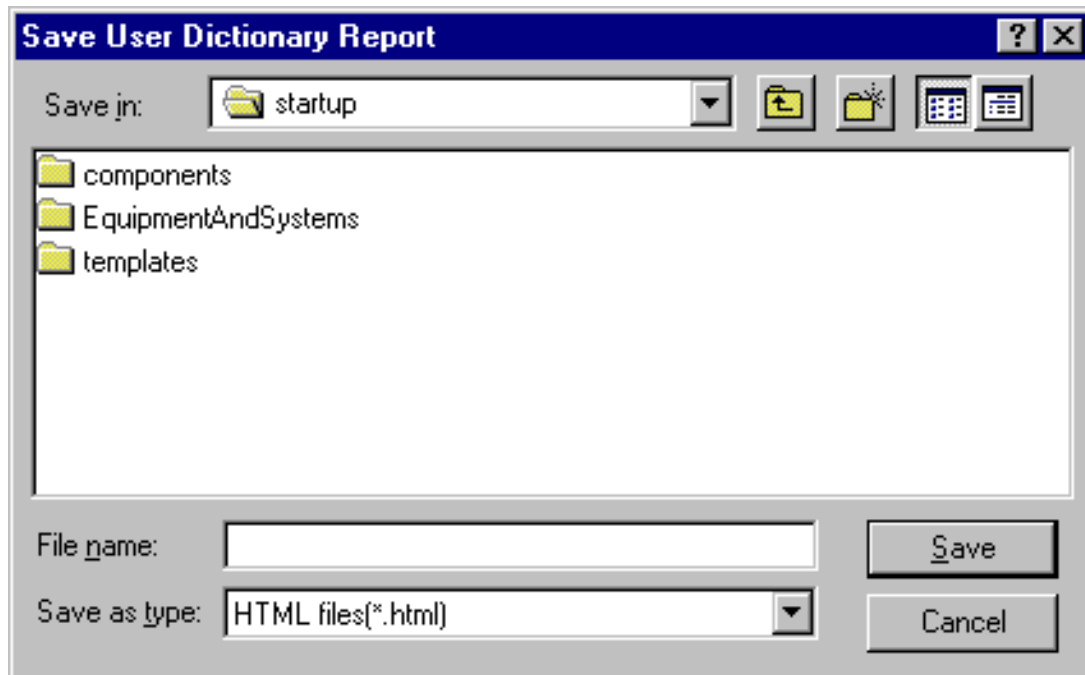
This task shows you how to generate a report that displays all objects.



This function allows you to generate an external file. The report will display all the objects under the Classes View and the Referenced View of the specifications tree. It will also display all the attributes associated with an object. See [Starting the Feature Dictionary Editor](#).



1. Click the **Generate Report** button . The **Save User Dictionary Report** dialog box will display.



2. Navigate to the directory where you want to save the file, give the file a name and save it. The report will be saved in HTML format.

3. To view the report open it in your Web browser.

User Dictionary Report

Dictionary file: CATfct1.CATfct

[Piping Part Function](#)

- [Branch Function](#)
- [In-line Instrument Function](#)
- [Reducer Function](#)
- [Valve Function](#)
- [newvalve](#)

[Pipe Function](#)

[Piping Line](#)

[Piping Part](#)

- [Branch](#)
- [Elbow](#)
- [Pipe](#)

 - [Straight Pipe](#)
 - [Bendable Pipe](#)
 - [Curved Pipe](#)

- [Fitting](#)
- [Flange](#)
- [Gasket](#)
- [Valve](#)

[Spool](#)

Piping Part Function

Inherited	User	Type	Va
-----------	------	------	----

- To view the attributes click on any of the objects. The report will display both inherited and local attributes.

newvalve

Inherited Attributes	User	Type	Value
Nominal size	System	String	<discrete>
Pipe specification	System	String	<discrete>
Nominal size_2	System	String	<discrete>
NominalSize3	System	String	<discrete>
NominalSize4	System	String	<discrete>
Current valve position	System	String	<discrete>
Normal valve position	System	String	<discrete>
Local Attributes	User	Type	Value
valveattribute	User	Length	123mm

[Back to Top](#)

Pipe Function

Inherited Attributes	User	Type	Value
none			



Creating a New Feature Dictionary

P2



This task shows you how to create a new Feature Dictionary.



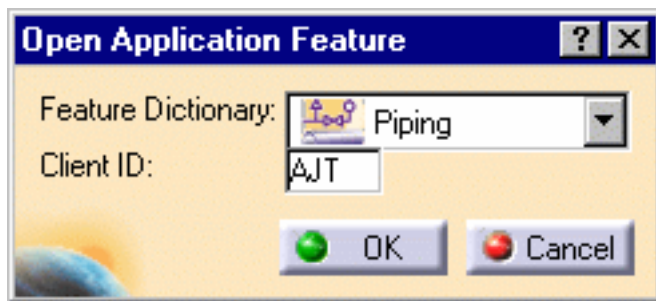
Many users will find it sufficient to add classes or attributes to the sample feature dictionary provided with the application. Some may prefer to create one or more new dictionaries. To be able to use a new feature dictionary you must change [settings](#).



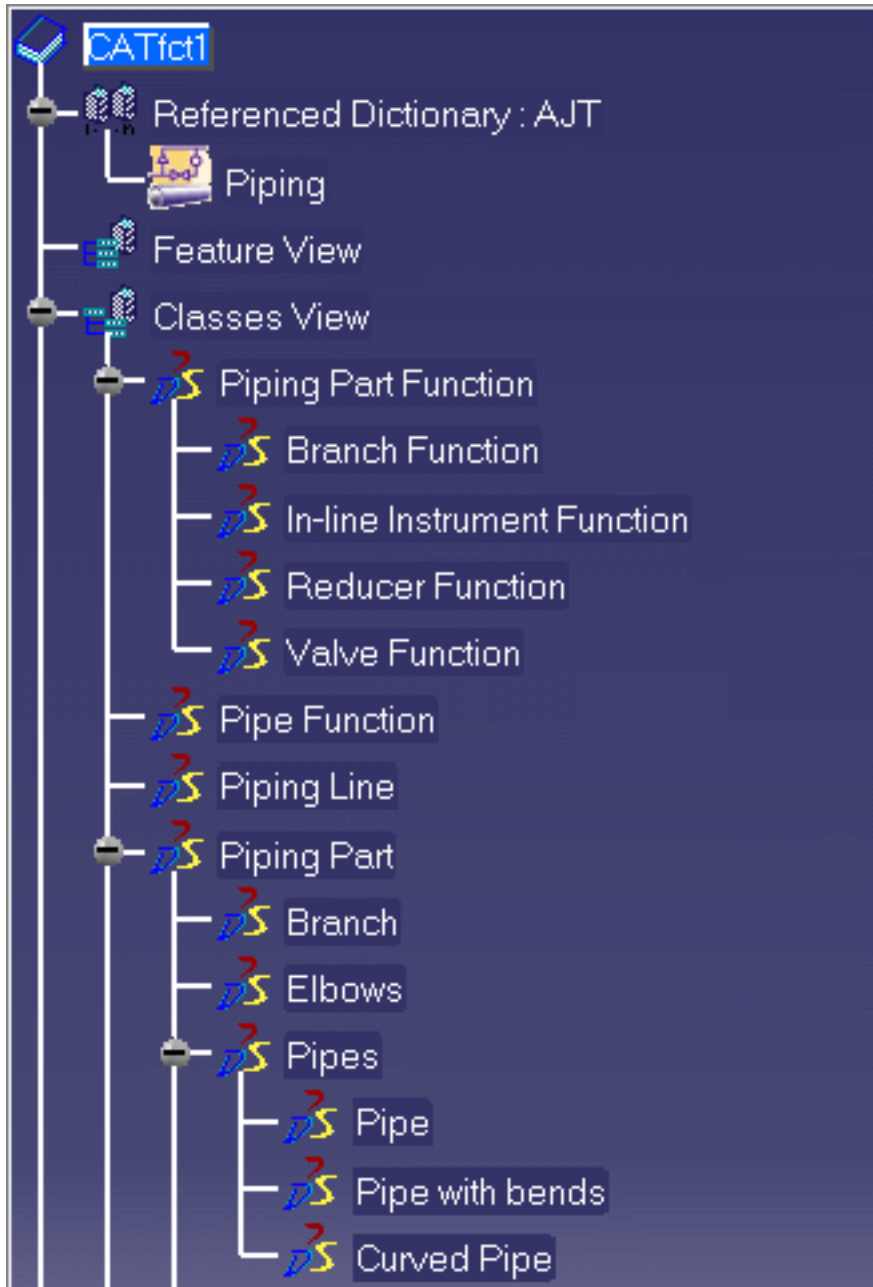
1. Start the feature dictionary editor and click the **Open Application Dictionary** button



The Open Application Dictionary dialog box opens.



2. Click the down arrow to select an application, such as Piping, and enter a Client ID. Click OK. A new .CATfct is created and the basic classes available to you appear in the feature editor.





3. Make your additions and save the .CATfct file to the directory:
intel_a\resources\graphic.



Opening a Reference Dictionary

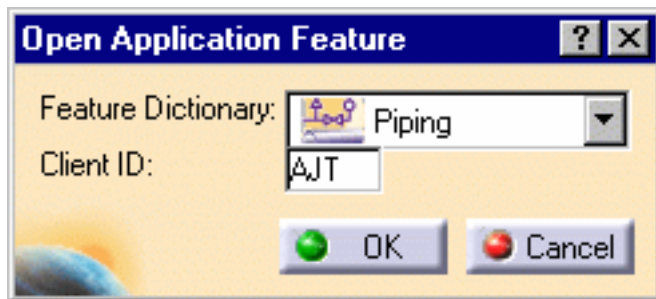
P2

 This task shows you how to create a open a feature file of a different domain from your current document to the Feature Dictionary Editor for viewing purpose.

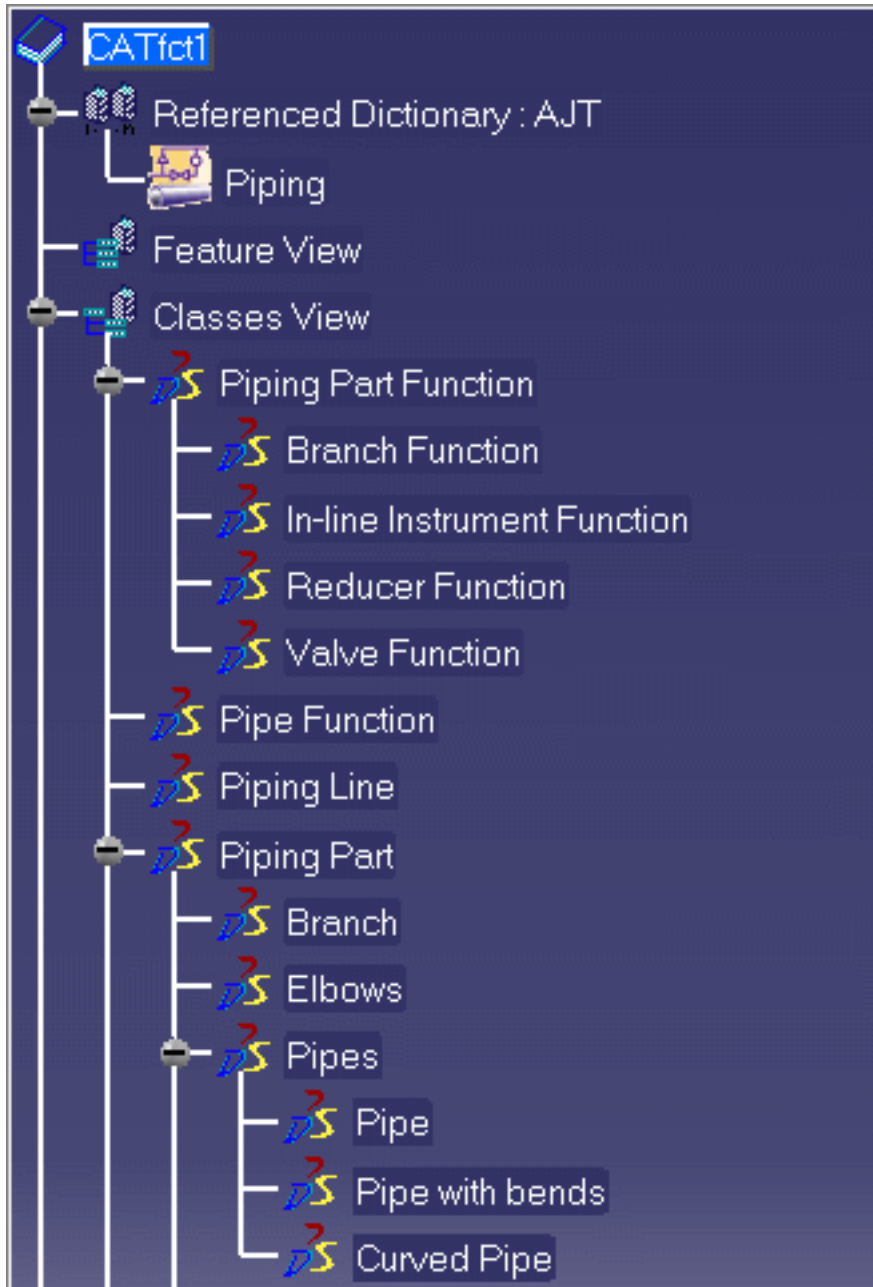
 **1.** Start the feature dictionary editor and click the **Open Application Dictionary** button



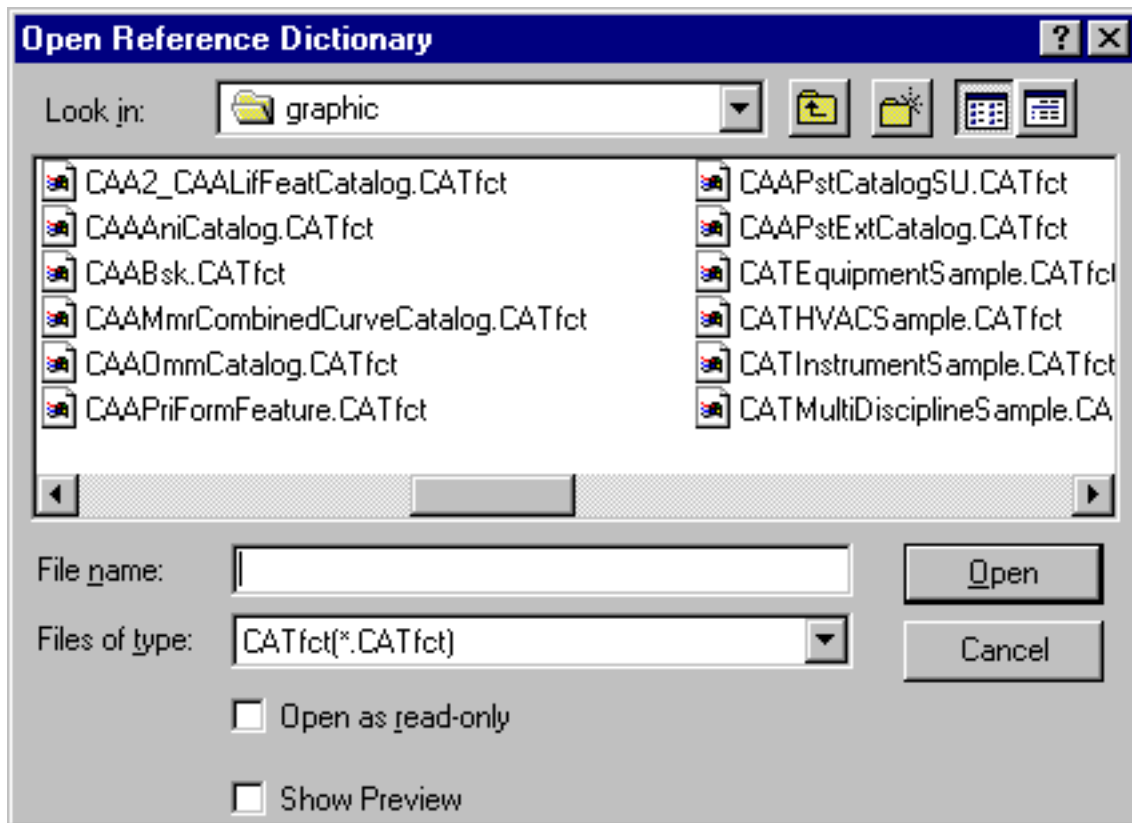
The Open Application Dictionary dialog box opens.



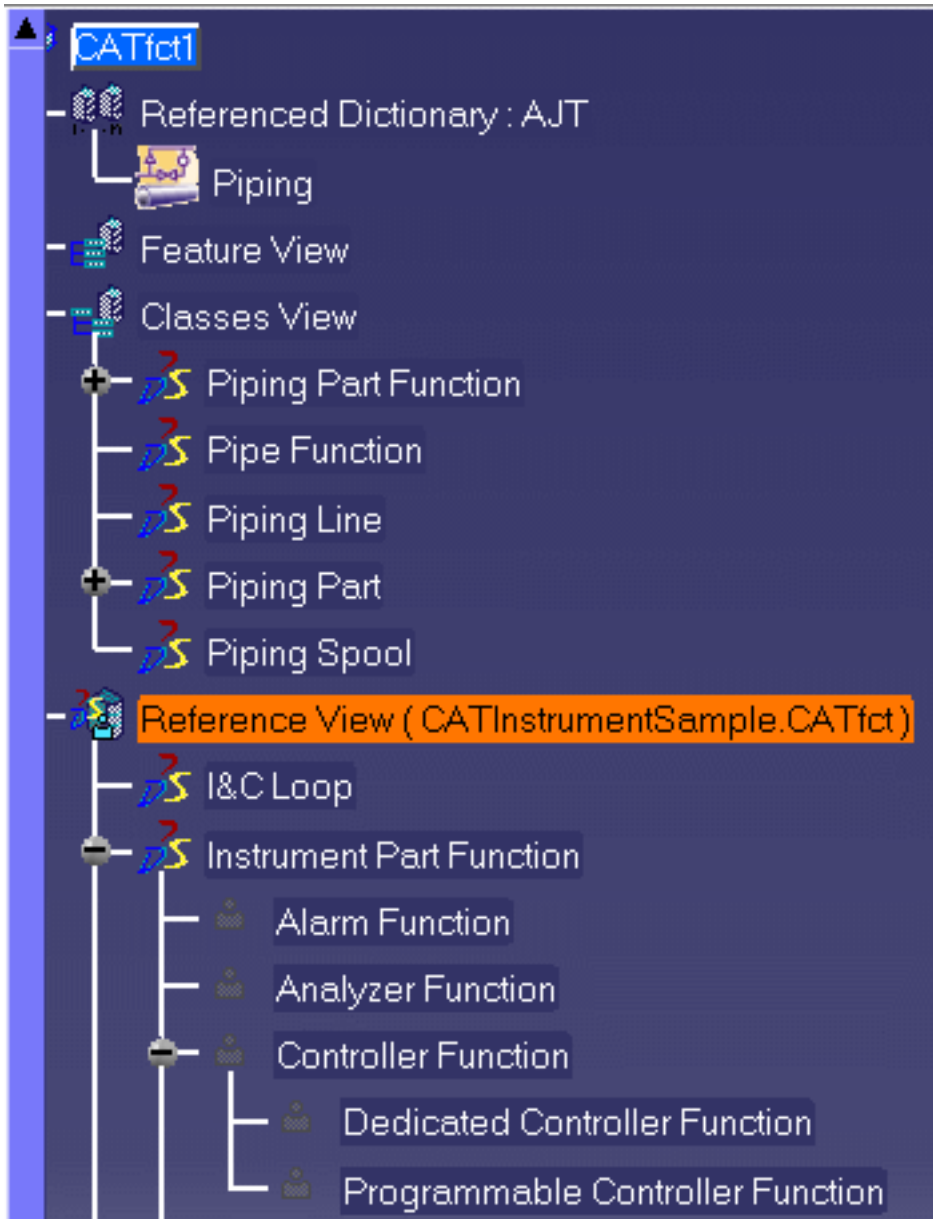
2. Click the down arrow to select an application, such as Piping, and enter a Client ID. A new .CATfct is created and the basic classes available to you appear in the feature editor.



3. Click the **Open Reference Dictionary**  button. The Open Reference Dictionary dialog box opens:



4. Navigate to select the .CATfct or .feat file you wish to open then click **Open**.



As you can see it above, the content of the reference dictionary you selected is displayed under the Reference View in the specification tree.

The document from the Piping domain is still the active document while the reference dictionary from the Instrument domain is displayed for viewing purpose.

i You can add attributes to the object that is under the Reference View. However, creating subclasses is forbidden.



Modifying the Object Naming Rules



This task shows you how to modify or define the object naming rules.



Examples from the Piping Design product are used in this task. The procedure is the same for all products that have this function - substitute the appropriate file or object when using another product.

Every object that you create (except a run), or part that you place, in your design document can be given a unique identifier. This identifier usually consists of a prefix that identifies the type of object or part it is, followed by a unique number. This enables users, for instance, to maintain a history of each part - when it was serviced, or repaired or replaced - and schedule servicing and replacement dates.

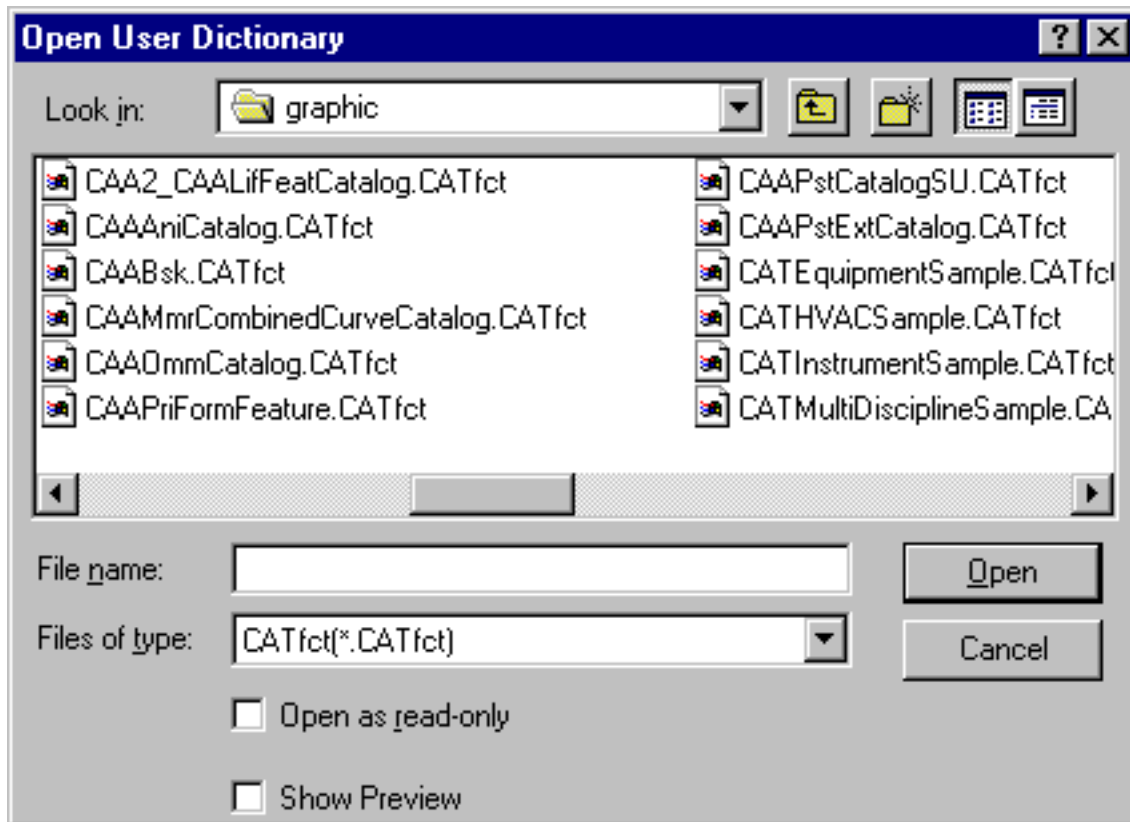
When you create an object or place a part in your document the application will suggest a name for it - the default name. (In many cases you have the option of rejecting this name and entering a different name, or renaming it.) The default name is based on certain rules. A set of default rules is included with this application, but most users will want to modify these rules to suit their own requirements. You can modify or define the naming rules in the following way:



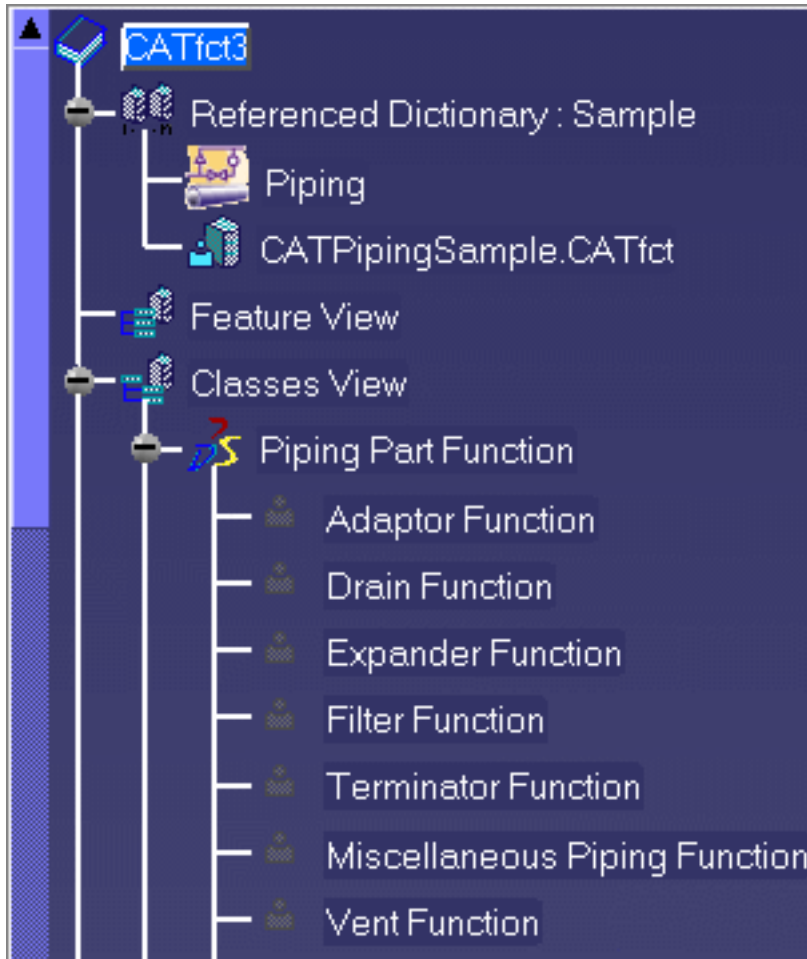
1. Open the Feature Dictionary Editor by selecting **Start->Infrastructure->Feature Dictionary Editor**.




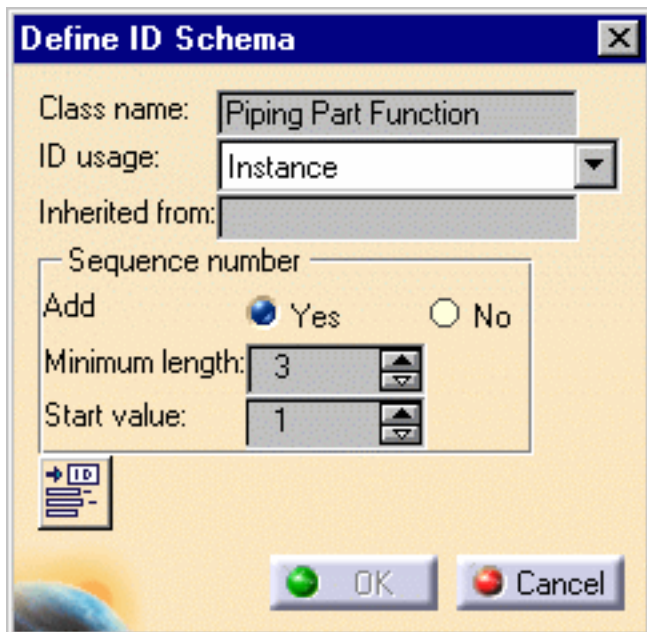
2. Click the **Open User Dictionary** button . The Open User Dictionary dialog box displays.



3. Navigate to the directory where your .CATfct files are stored. The default is ..
intel_a\resources\graphic. The CATfct files contain a list of all the object classes. Select and open the file associated with the product you are working with, i.e. Piping or Tubing, etc. All the classes in the file are displayed in the Feature Dictionary.



4. Select a class in the specifications tree and click the **Define ID Schema** button . The **Define ID Schema** dialog box will display.

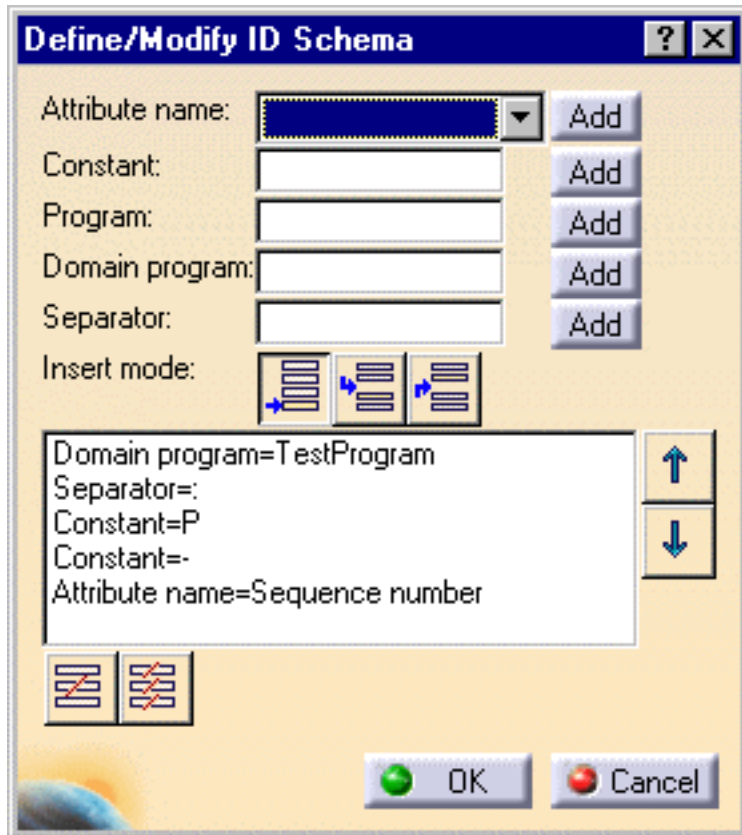


5. You have two options In the **ID usage** field, Instance and Reference, and you usually have to define naming rules for each object using both options. The naming rules you define under the Instance option are used by the application when you are placing a part in a document.

The naming rules you define using the Reference option are used by the application when you build a component for placing in a catalog. Most users will define naming rules for an object using both options. Depending on your needs, you can choose to simplify the procedure by defining rules for the parent function, which is Piping Part Function in the example above, and these rules will be inherited by all the objects under it.

If the **Inherited from** field is blank it means the name is not inherited from another class. Select *Yes* or *No* for **Sequence number**. Yes or No cannot be selected if you have Reference as the ID usage. **Minimum length** refers to the number of digits in the numbering scheme. For instance, 3 means the number will show up as 001.

6. Click the **Define/modify ID schema** button . The **Define/modify ID schema** dialog box will display.



7. In this dialog box you can define what you want to appear in the name of an object, in this case the object being Piping Part Function. The dialog box has a window in the lower half which displays the current naming scheme. You can delete one or more of the fields using

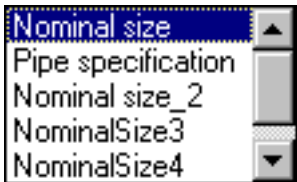


the Delete field/Delete all fields buttons

Click **Add** after entering or selecting a value in a field. You can choose to have more than one attribute value displayed in a name, for instance when you want to add a Separator at more than one place. Click Add after selecting each one.

You can select the order in which the values will appear in a name by using the Up or Down arrows or the buttons in the **Insert mode** field.

8. Select one of the attributes from the drop down list in the **Attribute name** field.



If you select **Nominal size**, for instance, the object name will display the nominal size of the object. These attributes are for the Piping Part class only - other classes will have different attributes displayed. You can display more than one attribute in the name.

9. Enter any value you want displayed in the **Constant** field. If you enter PP (for Piping Part), all piping part names will display this value. You can add a constant to a name anywhere you require it. For instance, you may begin a name with PP, and end it with WR for a project name.

- 10.** The **Program** field is used to execute a program that will then add a value to the name. You can create your own programs, but some sample programs are provided with the application and are listed below. Enter a program name in this field if you want it to be executed. For instance, if you enter **CATPspEncSchedule** in the field, then the *short value* of the Encoded Schedule attribute will be added to the name (the short value of *Extra Strong* is *XS*). These programs are Standards-based and will execute based on the standard you have defined in your Options. The default standard is ASTL.

The following list shows the programs provided with the application as a sample, and the attributes they refer to:

- **CATPspEncRating** - Encoded Rating
- **CATPspEncRating2** - Encoded Rating2
- **CATPspEncRating3** - Encoded Rating3
- **CATPspEncRating4** - Encoded Rating4
- **CATPspEncNominalSize** - Encoded Nominal Size
- **CATPspEncNominalSize2** - Encoded Nominal Size2
- **CATPspEncNominalSize3** - Encoded Nominal Size3
- **CATPspEncNominalSize4** - Encoded Nominal Size4
- **CATPspEncSchedule** - Encoded Schedule
- **CATPspEncMaterialCategory** - Encoded MaterialCategory
- **CATPspEncMaterialCode** - Encoded MaterialCode

- 11.** The **Domain program** field is used to execute a program that will add the name of the domain to which the object belongs. Domain in this case refers to an object to which the object to be named is connected. For instance, when naming a nozzle it is preferable to add the name of the equipment to which it is connected. One sample domain program is provided with the application, and provides this function: **CATPspConnectedEquip**.

- 12.** The **Separator** field is used to add separators, such as a hyphen or semi colon, after the domain field.

13. Use the buttons in the **Insert mode** field to organize the name. **Append field to list** will move a field to the end of the name. The other buttons are used when you are adding a field, to position it in the name.



Using the Data Upward Assistant



The **CATDUA V5** tool, also called **Data Upward Assistant**, allows you to have a support for CATIA level changes, to make a diagnostic, and eventually a healing of CATIA Version 5 data. If some return codes are detected within CATIA elements, it is advisable to run the Data Upward Assistant utility on these documents in order to check and/or upgrade them. And for better performances, the Data Upward Assistant and CATIA versions should be the same.

When is the Data Upward Assistant useful?

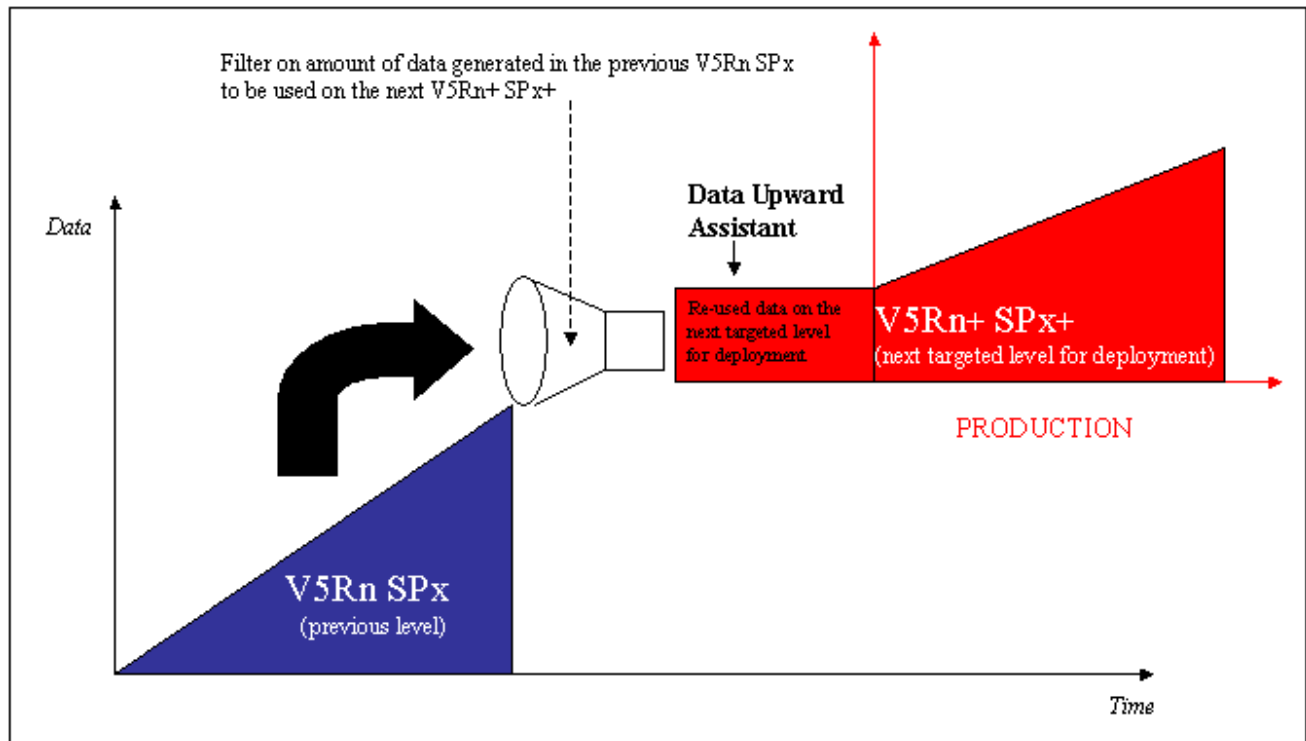
- before recovering external data
- before going into a new CATIA release
- broken links when opening CATProducts
- incidents when updating a component (for instance, Sketch update)
- the Edit-Links panel appears: some documents are found but they have no references.
- performance problems when opening a CATProduct (because some elements have lost their links).



The CATDUA V5 is also available from **ENOVIA DMU**.

For ENOVIA DMU, the Data Upward Assistant is P2 only.

Here is a schema explaining when it should be used:



The following tasks explain you how to launch the Data Upward Assistant in interactive mode or in batch mode, how to visualize the report of the check/upward execution and lastly, how to interpret the return codes that can be detected by the Data Upward Assistant.

[Using CATDUA V5 in Interactive Mode](#)

[Using CATDUAV5 Batch](#)

[Viewing Results of CATDUA V5 Execution](#)

[Return Codes Detected by the Data Upward Assistant](#)

[List of the Return Codes](#)

Using CATDUA V5 in Interactive Mode



This task will show you how to launch and use the CATDUA V5 (also called Data Upward Assistant) out of or in a CATIA V5 session.

This assistant allows you to make a diagnostic and eventually to heal a CATIA document (CATPart, CATProduct, CATProcess, CATAnalysis, CATDrawing). A report of check/upward result is available after using the Data Upward Assistant.

- [Data Upward Assistant out of a CATIA session](#)
- [Data Upward Assistant in a CATIA session](#)



The Data Upward Assistant settings work with the same CATIA settings. If the **Cache Activation** option (in CATIA **Tools->Options->Infrastructure->Product Structure**) or the **Load referenced documents** option (in CATIA **Tools->Options->General->General**) have been selected, they will be taken into account in the Data Upward Assistant.

In CATIA **Tools->Options->General->General**, it is recommended to select the **Load referenced documents** option.

Data Upward Assistant out of a CATIA session

On Windows

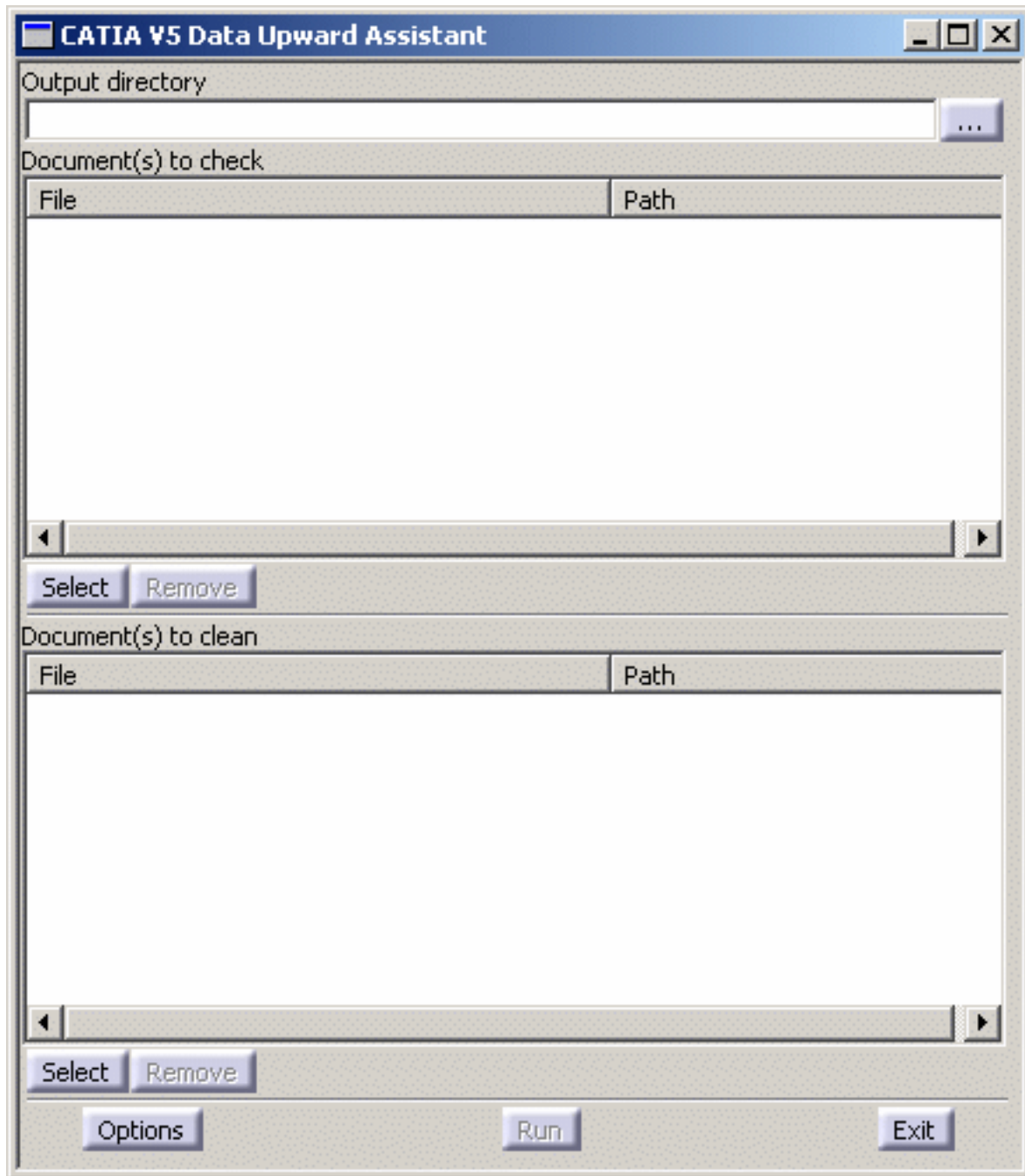


1. In a MS-DOS window (or Command Prompt), go to the level in which CATIA is installed (example: E:\Install...).

2. Enter the following command:
`cd intel_a\code\bin`

3. Enter the following command:
`CATDUAV5`

The CATIA V5 Data Upward Assistant dialog box appears.



On UNIX



1. Change the directory to:
cd Install_folder/OS_a/code/command

2. Run the following command:

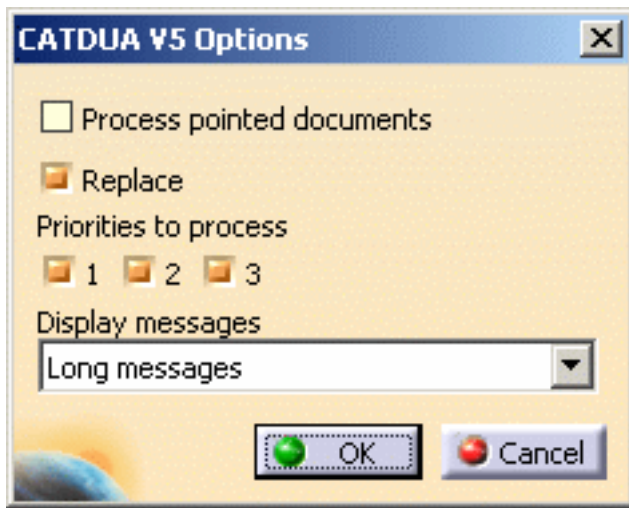
```
./catstart -env EnvName -dirEnv DirName -run "CATDUAV5"
```

Note that **EnvName** is the environment file and **DirName** is the directory in which the environment is.

The same CATDUAV5 dialog box appears.


Description of the CATIA V5 Data Upward Assistant dialog box

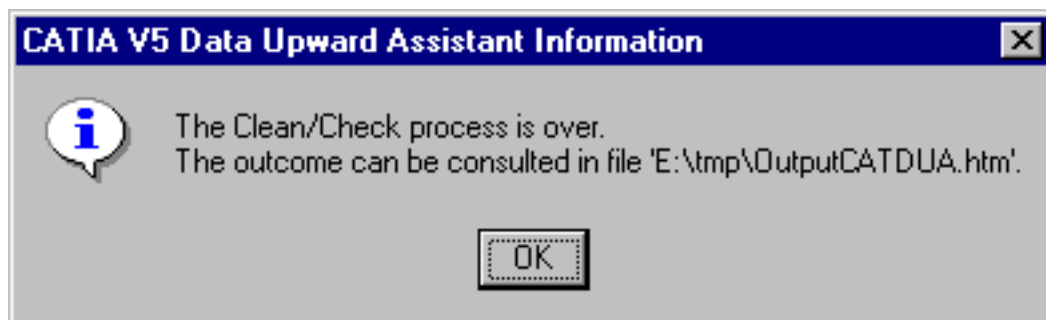
- **Output directory:** this field contains the path directory indicates where the upgraded document will be saved.
- **Browse :** this button lets you select the output directory in which the CATIA upgraded documents will be saved. When you click this button, the Browser dialog box appears to select the desired directory path.
- **Document(s) to check:** this field lists all the selected documents you want to check.
- **Document(s) to clean:** this field lists all the selected documents you want to upgrade.
- **Select:** select the documents you want to replace, check or clean.
- **Check:** this button lets you select the CATIA documents you want to check. When you click this button, the Browser dialog box appears to select the documents you want to check.
- **Clean:** this button lets you select the CATIA documents you want to upgrade. When you click this button, the Browser dialog box appears to select the documents you want to upgrade.
- **Remove:** this button lets you deselect a CATIA document you have previously selected.
- **Options:** this button opens the CATDUA V5 Options dialog box.



- **Process pointed documents:** this option lets you choose if you want to check or upgrade all the linked components of a CATProduct. This option is taken into account only if you activate it before selecting the CATProduct. It is not applied to the previous selected documents.
- **Replace:** to save the processed document in the target directory you specified, check the Replace panel option. This means that if a document with the same name already exists in the target directory, this option will automatically replace the old document by the new one.
- **Priorities to process:** this option lets you choose the result of priority errors you want to get (1 and/or 2 and/or 3 choices).
- **Display messages:** this option lets you choose the kind of information you need in the report (**Long messages**, **Short messages**, **Short/Long messages**).

- **Launch:** this button lets you start the Data Upward Assistant execution.

 At the end of the check/upward process, a CATIA V5 Data Upward Assistant Information message appears to give you the path directory of the check/upward report.



For more details about check/upward report, see [Viewing Results of CATDUA V5 Execution](#).



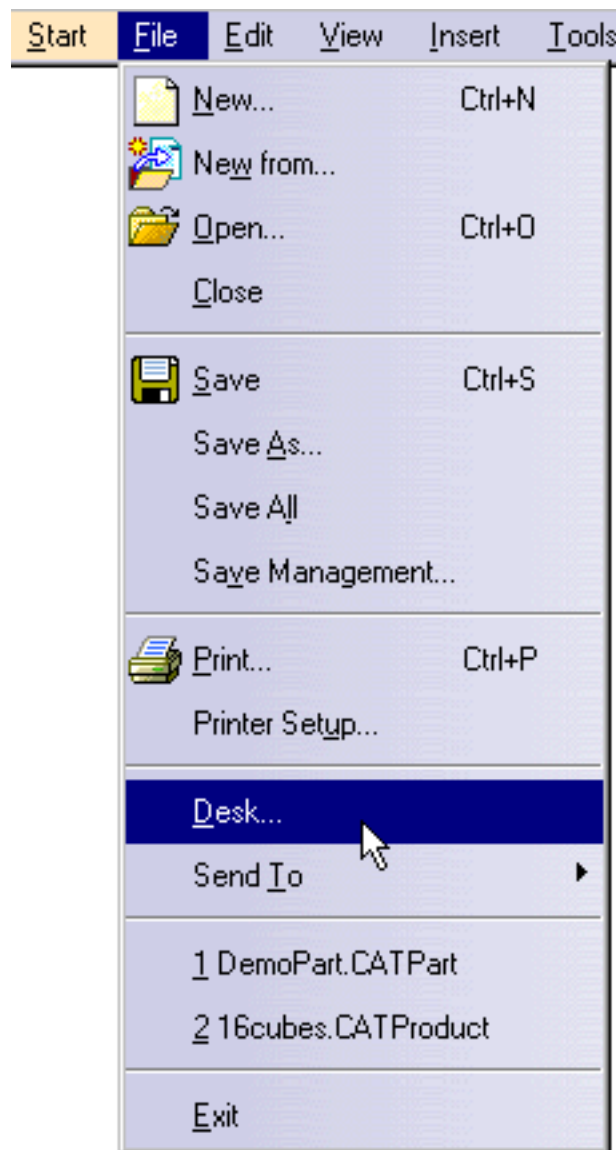
Data Upward Assistant in a CATIA session



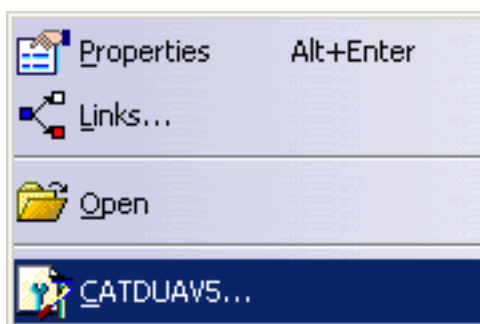
- The Data Upward Assistant in a CATIA session allows you to check or upgrade only file documents from CATIA V5 (documents which have been saved in a CATIA V5 session).
- The Data Upward Assistant in a CATIA session allows you to check or upgrade only one CATIA document and not the pointed documents.
- After checking and upgrading your document, you have to save it, if desired. It is not automatically done.
- The htm result is automatically saved in a subdirectory (**CATDUAV5UI**) of the default directory CATTemp of CATIA.



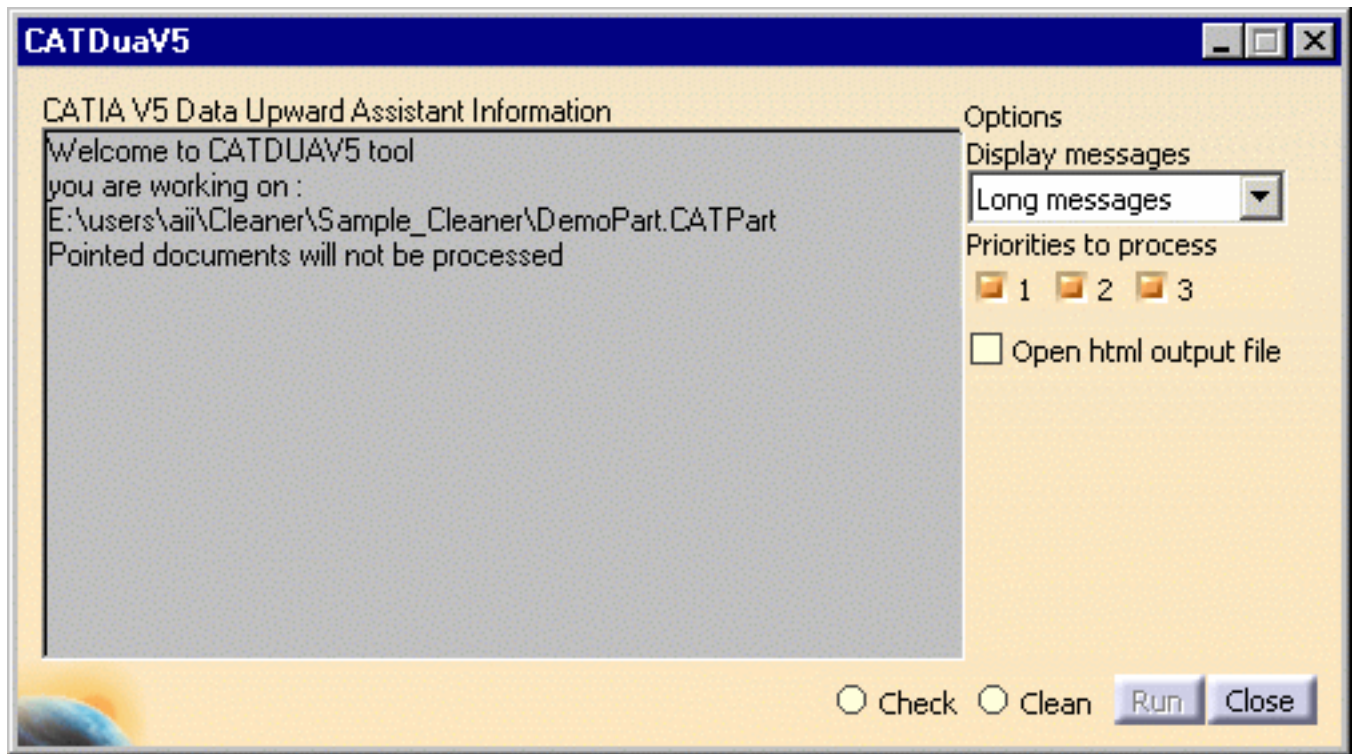
1. Select the **Desk...** submenu in the **File** menu.



2. Right-click the CATIA document you want to check or upgrade and select the **CATDUAV5...** contextual menu.



The CATDuaV5 dialog box appears.



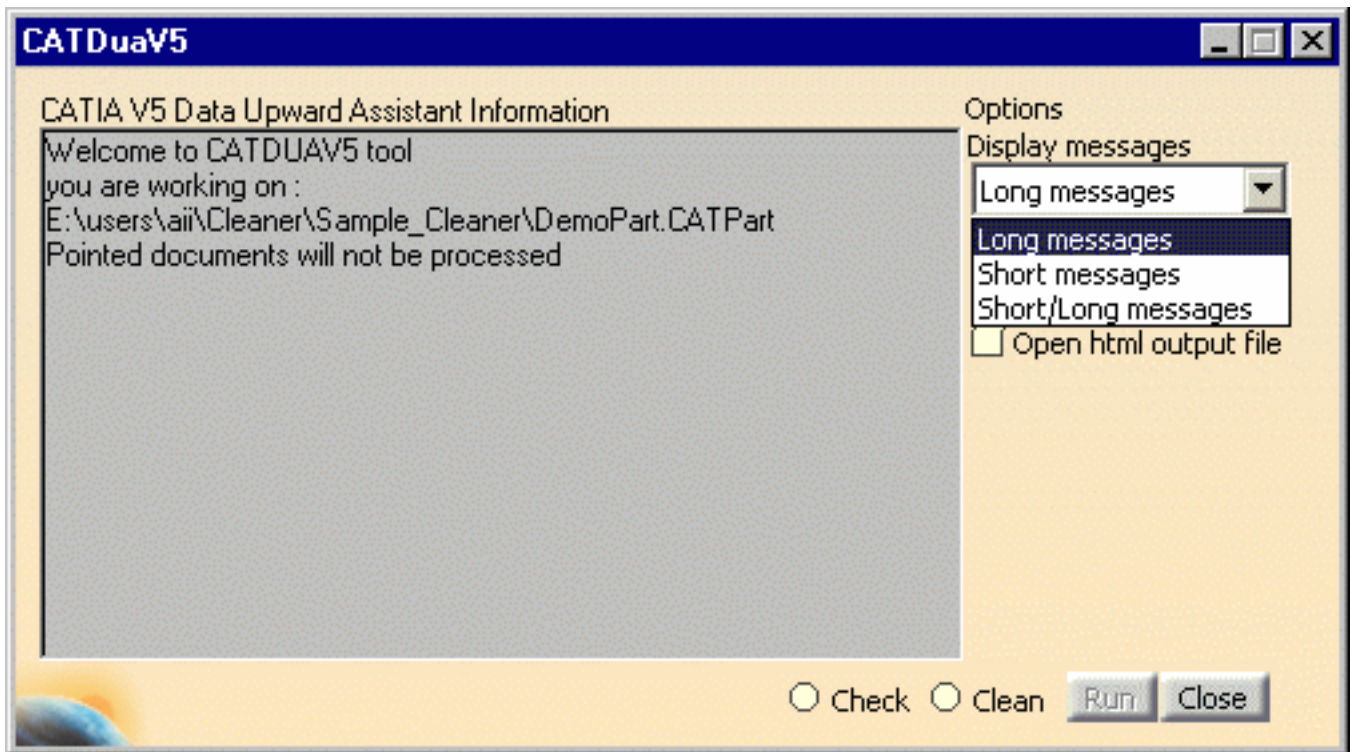
Description of the CATDuaV5 dialog box

- **CATIA V5 Data Upward Assistant Information:** this field displays information about the CATIA document you are working on and several informative messages.
- **Options:**
 - **Priorities to process:** lets you choose the result of priority errors you want to get (1 and/or 2 and/or 3 choices).
 - **Display messages:** lets you choose the kind of information you need in the report.
 - **Open html output file:** the OutputCATDUA.htm file is immediately opened when the Check or Clean process is finished. For more information about this result report, please refer to [Viewing Results of CATDUAV5 Execution](#).
- **Check:** this mode of execution lets you check the CATIA document without correcting it.
- **Clean:** this mode of execution lets you check the CATIA document and to correct it.
- **Launch:** this button lets you start the check or upward execution.
- **Close:** this button lets you close the CATDuaV5 dialog box.

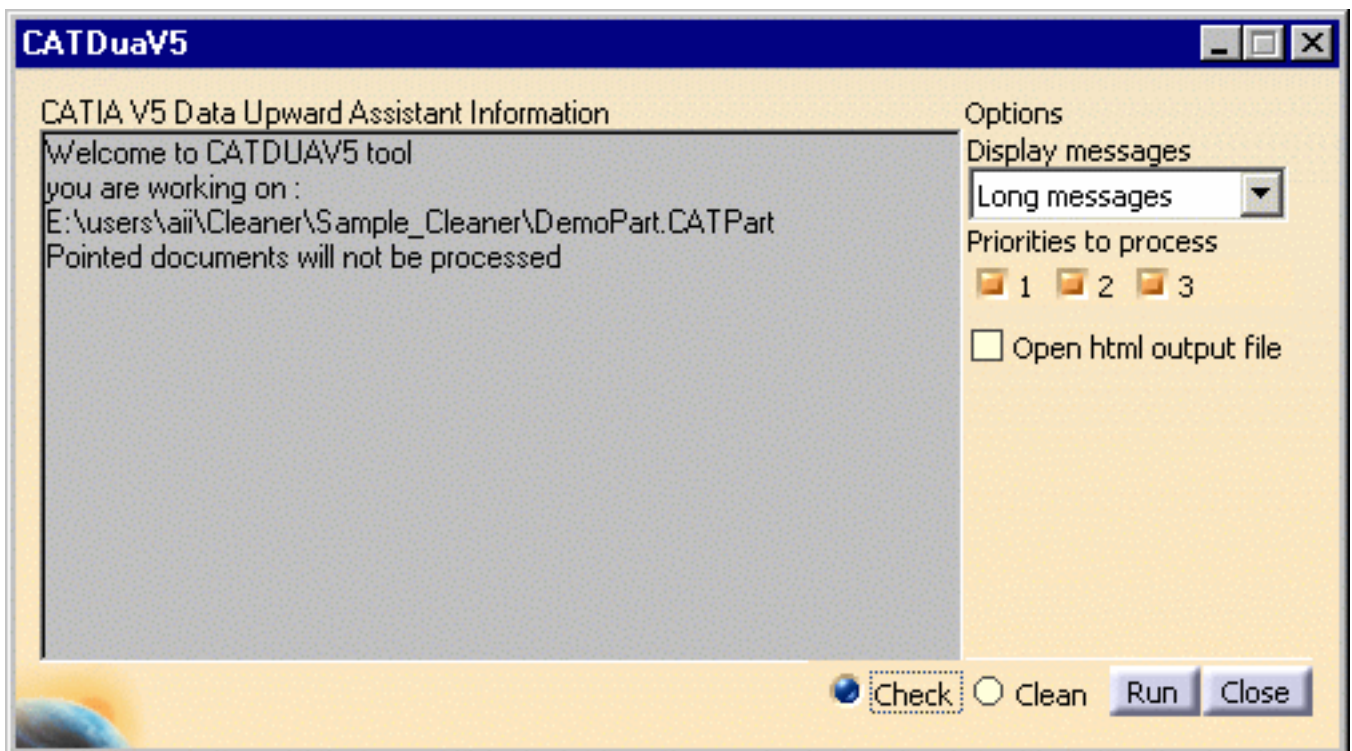
Check a CATIA document without correcting it



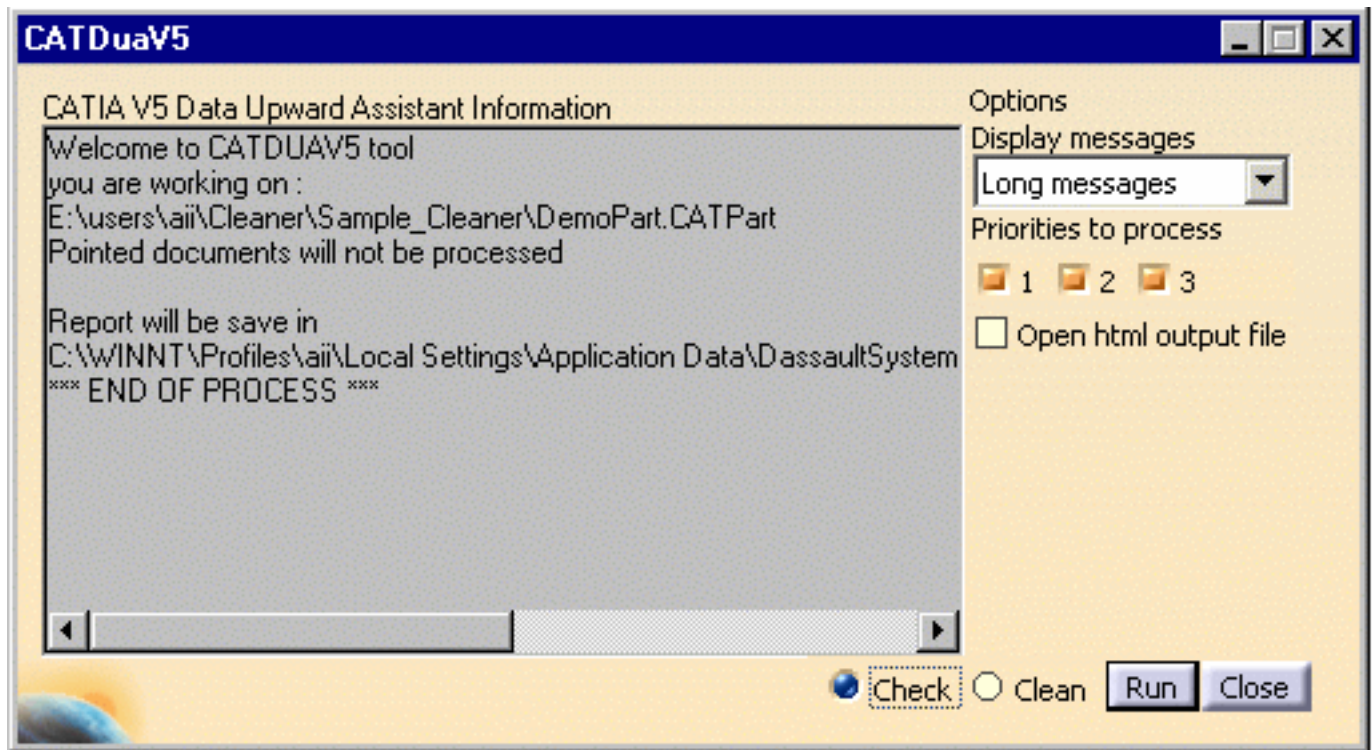
- 1.** Choose the level of priority in the **Priorities to process** field. You can choose one, two or all the options simultaneously.
- 2.** Choose the type of message you want to have.



3. Activate the **Check** mode.





4. Click the **Launch** button to start the CATDUA V5 execution.
A message appears in the **CATIA V5 Data Upward Assistant Information** field. It explains where the check results are.

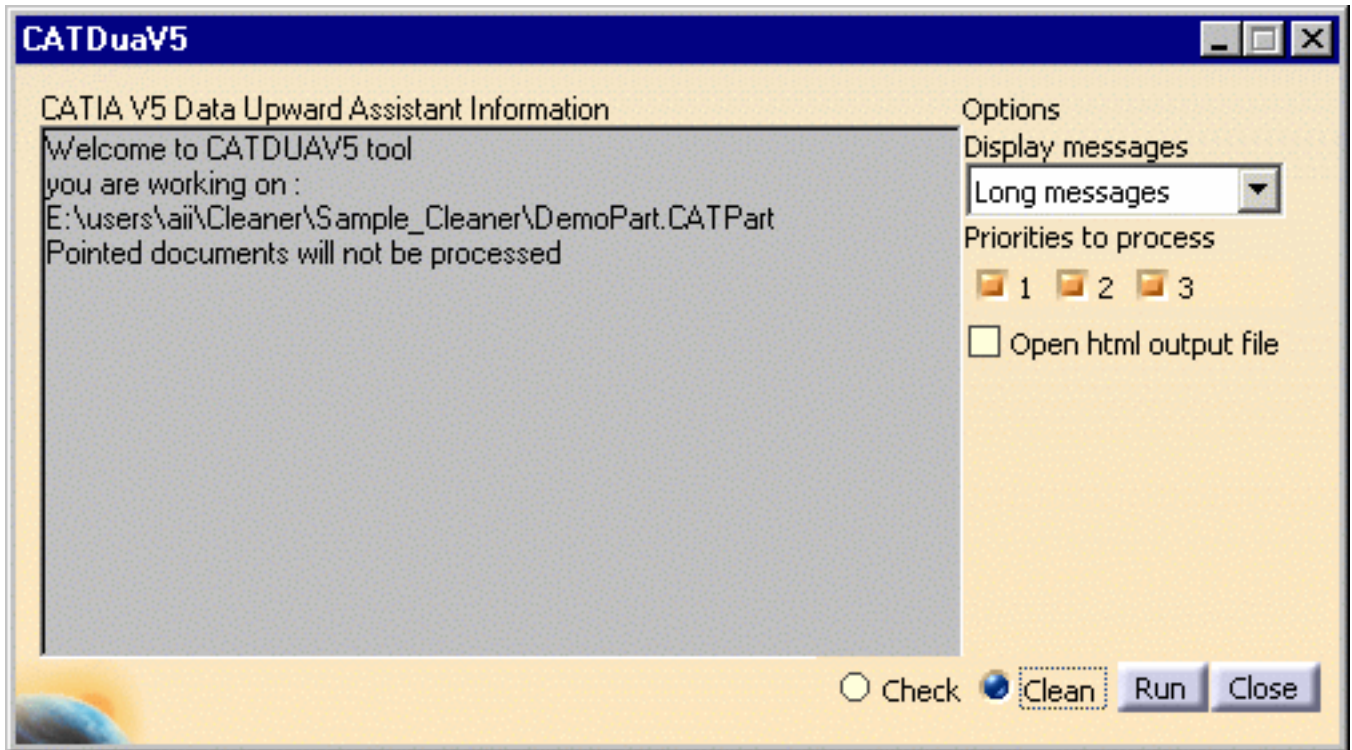


The CATIA V5 Data Upward Assistant Information field displays the path directory of the check/upward report.

Upgrade a CATIA document

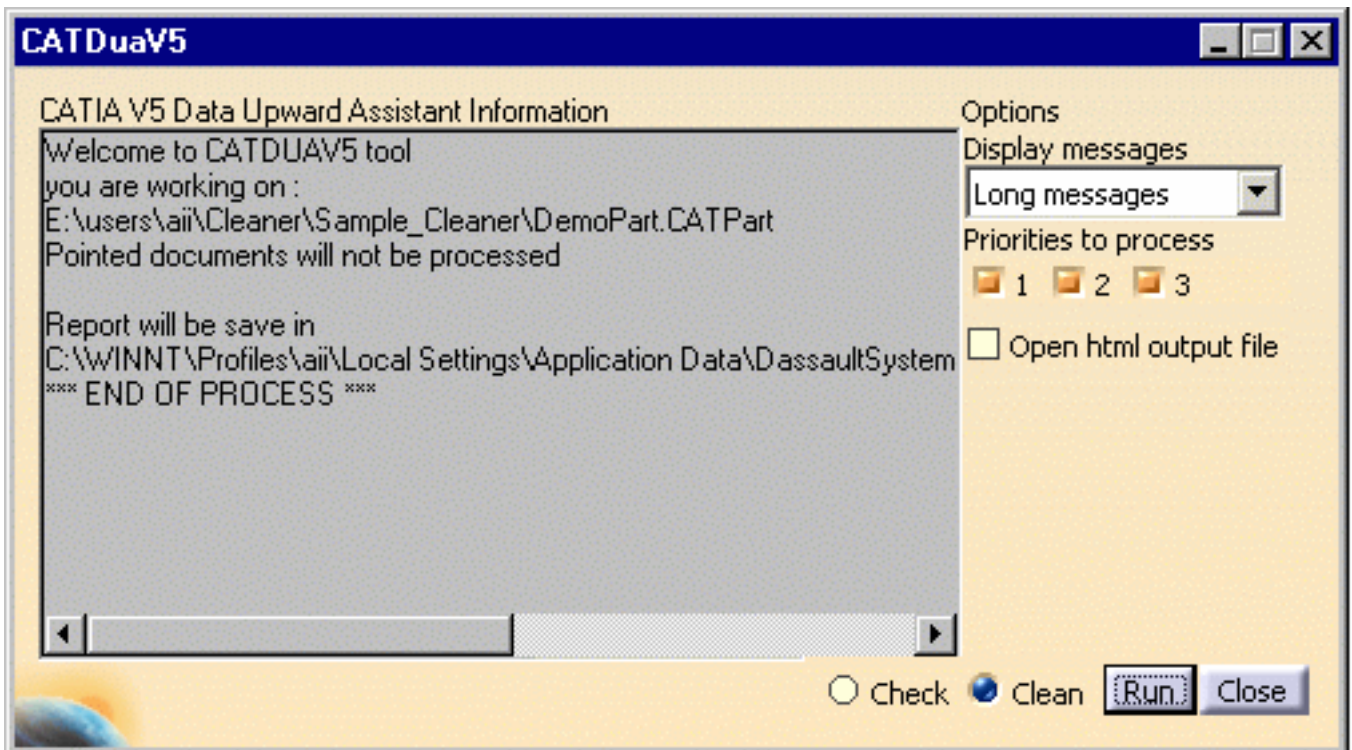
When you choose the Clean mode, the Data Upward Assistant checks previously the CATIA document.

-  Even if you have already executed a check, you can launch the upward execution without closing the CATDuaV5 dialog box.
- 
 - 1.** Choose the level of priority in the **Priorities to process** field. You can choose one, two or all the options simultaneously .
 - 2.** Choose the type of message you want to have.
 - 3.** Activate the **Clean** mode.



4. Click the **Launch** button to start the CATDUA V5 execution.

A message appears in the **CATIA V5 Data Upward Assistant Information** field. It explains where the upward results are.



The CATIA V5 Data Upward Assistant Information field displays the path directory of the check/upward report.

5. Save the upgraded document, if needed.





For more details about the check/upward results, see [Viewing Results of the CATDUA V5 execution](#).

Viewing Results of CATDUA V5 Execution



This task will show you how to visualize the report of a check/upward execution.

You can have access to the results of the check/upward execution when you use the Data Upward Assistant:

- in interactive mode
 - out of a CATIA session
 - in a CATIA session
- in batch mode



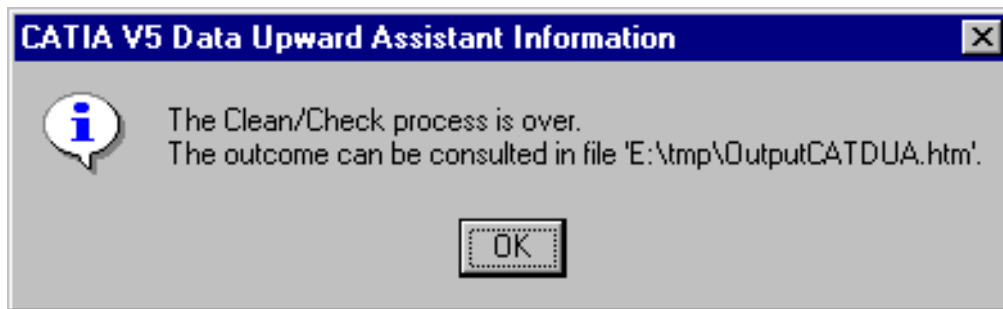
Before performing this task, you have to be familiar with the use of the Data Upward Assistant. For this, refer to :

- [Using the CATDUA V5 in Interactive Mode](#), in this guide
- [Using the CATDUA V5 Batch](#), in this guide

Interactive Mode

Out of a CATIA session

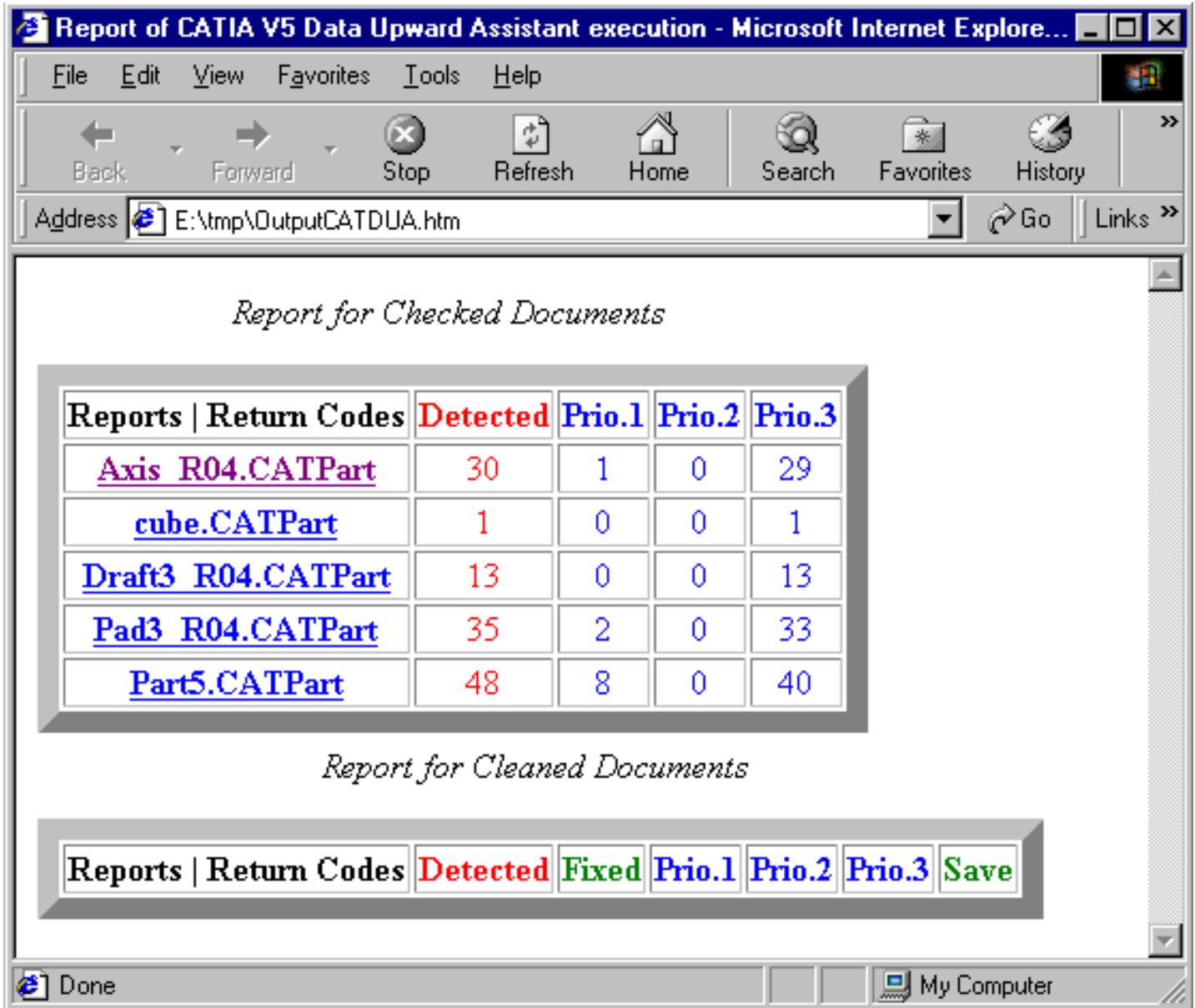
When you use the Data Upward Assistant out of a CATIA session, a message appears at the end of the check/upward process.



This message gives you the path directory of the check/upward report (path directory which you have defined above). The check/upward results are stored in a htm file (**OutputCATDUA.htm**). To have access to this file, you can use a Windows Explorer.

Example 1: In check mode on one or several CATParts

If you have selected more than one CATPart for the check execution, the htm file will give you the results for all the CATParts.



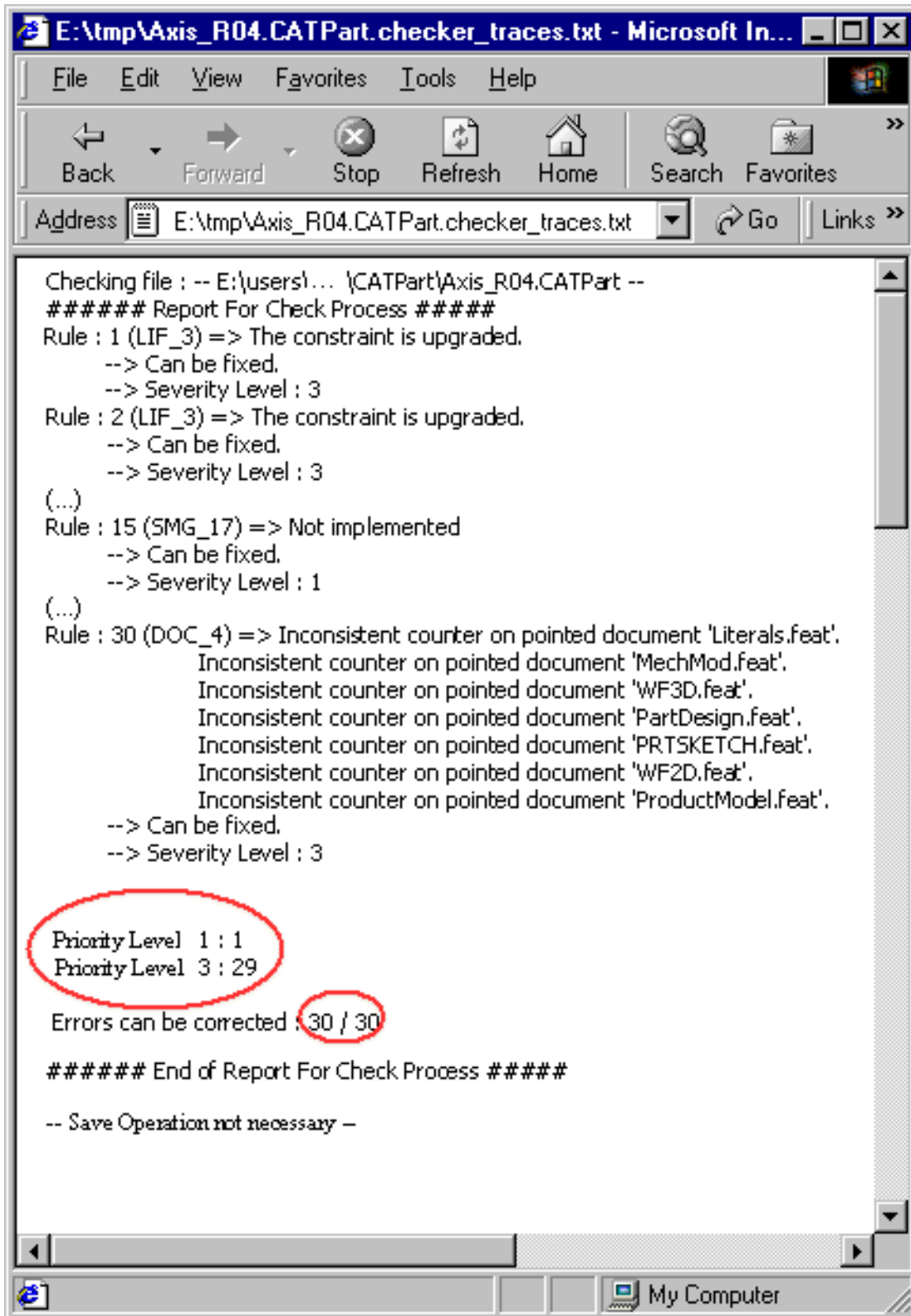
The output directory is not automatically emptied by the Data Upward Assistant before launching a new check or upward execution. If you do not wish to visualize your former results in the html file, you need to clear the directory. Moreover, the documents (CATParts, CATProducts, CATDrawings,...) in the output directory are not write-protected.

The delivered information is:

- name of the CATIA documents you have checked
- number of the detected return codes
- number of the fixed return codes
- number of the priority1 / priority2 / priority3 return codes
- hyperlinks pointing at a .txt file which give the details of the results for each CATPart (in the **Reports - Return codes** column)
- information about the save operation

You can have access to the txt file (...**checker_traces.txt**).
 For this, click on the hyperlinks in the **Report - Return Codes** column of the **Report for Checked Documents** table.

This document delivers the check results and informs you about the status of the return codes, if they can be repaired or not.

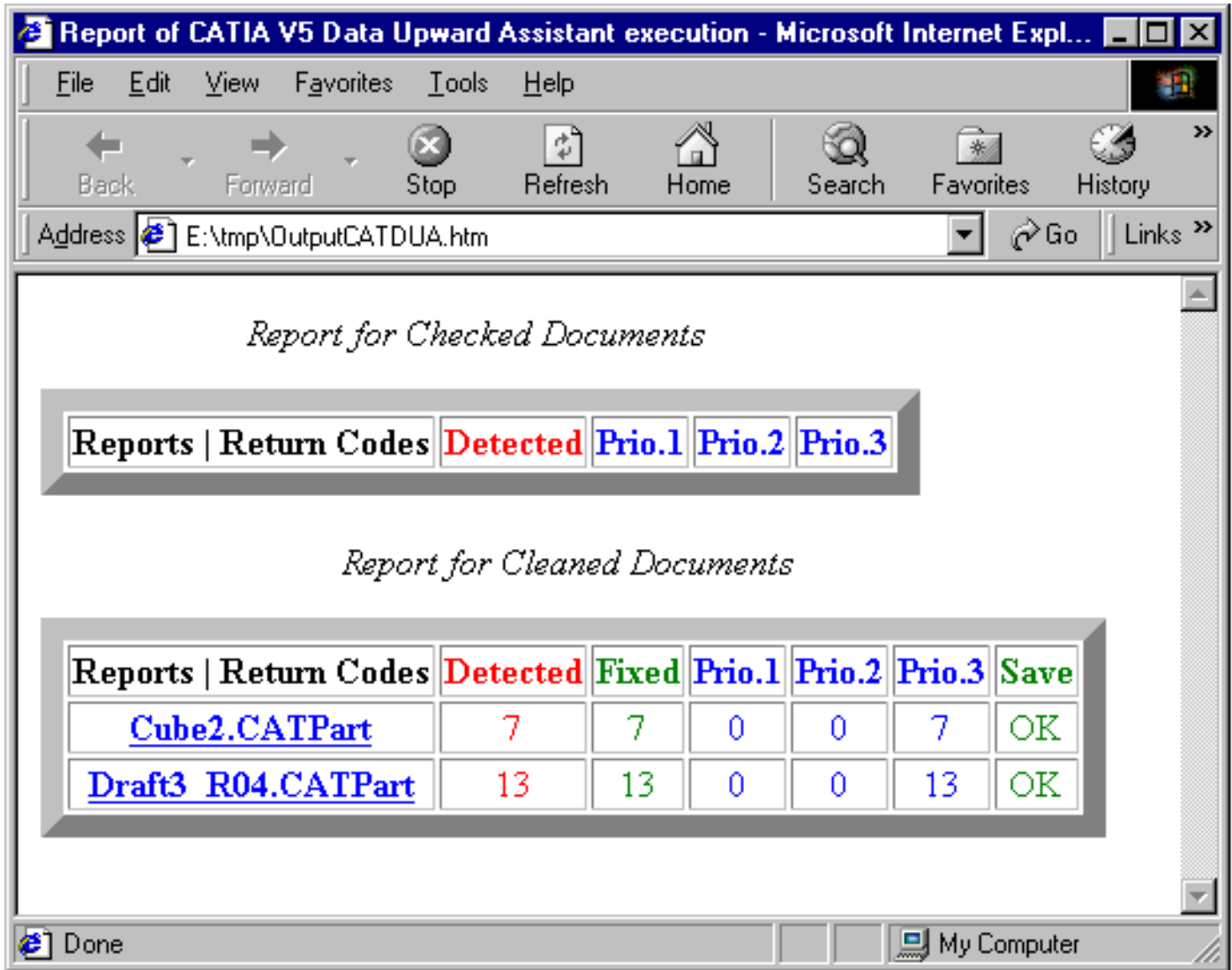


You can have access to:

- **Priority Level:** corruption level of the file.
- **Return codes can be corrected:** the number of return codes that can be solved in the upgrade mode.

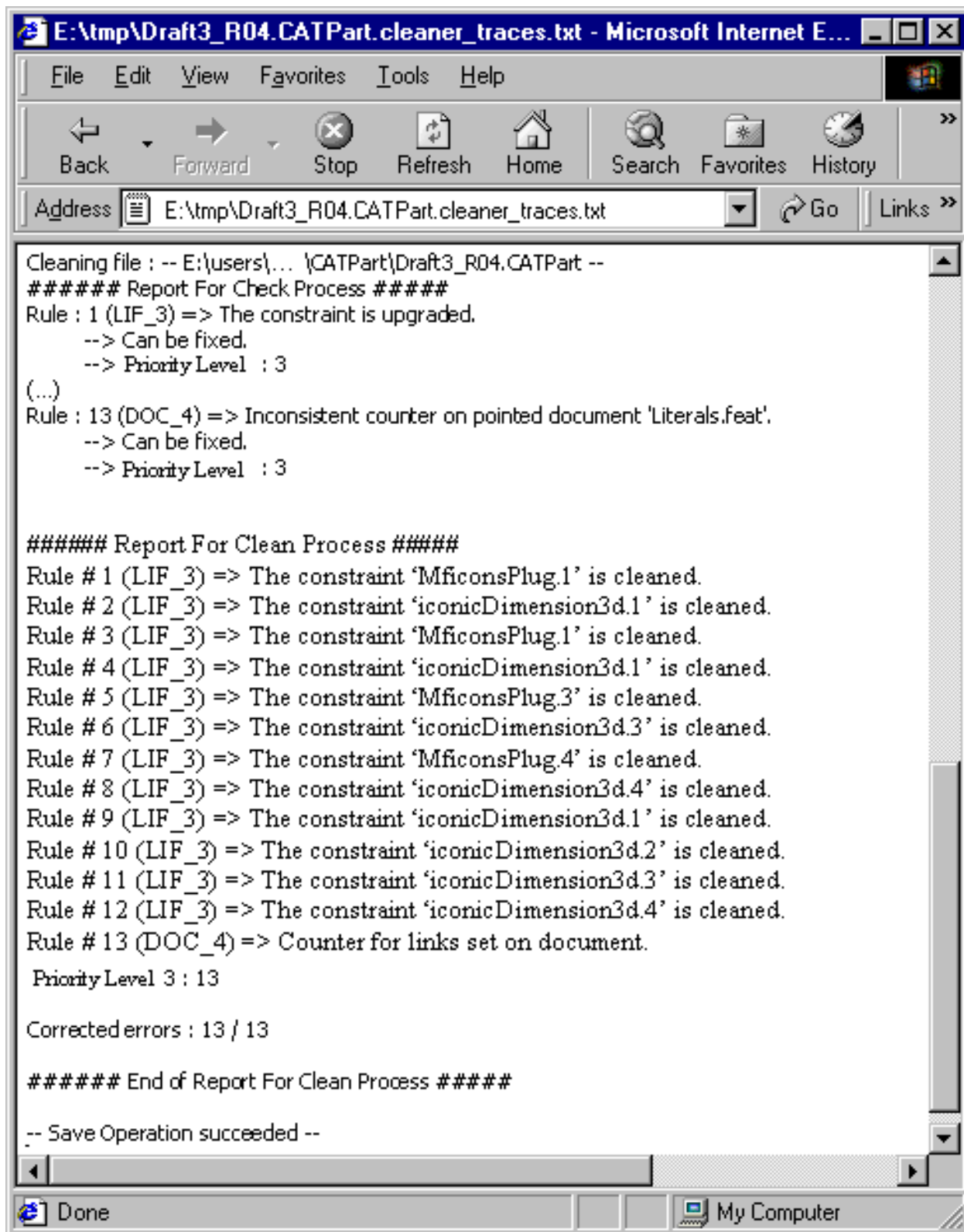
Example2: In upward mode

The file looks like the one for the check mode; this is an htm document, with a table entitled **Report for Cleaned Documents**, containing hyperlinks.



You can see **OK** in the **Save** column of the **Report for Cleaned Documents** table, that means the upgraded documents have been automatically saved. The saved document (CATPart, CATProduct,...) can be found in the same directory as the upward report file.

If you click on a link, you can find the upward results in the file**cleaner_traces.txt**, with the number of upward return codes out of the total number of return codes:



The screenshot shows a Microsoft Internet Explorer window with the address bar set to `E:\tmp\Draft3_R04.CATPart.cleaner_traces.txt`. The main content area displays a text file with the following text:

```
Cleaning file : -- E:\users\... \CATPart\Draft3_R04.CATPart --
##### Report For Check Process #####
Rule : 1 (LIF_3) => The constraint is upgraded.
    --> Can be fixed.
    --> Priority Level : 3
(...)
Rule : 13 (DOC_4) => Inconsistent counter on pointed document 'Literals.feat'.
    --> Can be fixed.
    --> Priority Level : 3

##### Report For Clean Process #####
Rule # 1 (LIF_3) => The constraint 'MficonsPlug.1' is cleaned.
Rule # 2 (LIF_3) => The constraint 'iconicDimension3d.1' is cleaned.
Rule # 3 (LIF_3) => The constraint 'MficonsPlug.1' is cleaned.
Rule # 4 (LIF_3) => The constraint 'iconicDimension3d.1' is cleaned.
Rule # 5 (LIF_3) => The constraint 'MficonsPlug.3' is cleaned.
Rule # 6 (LIF_3) => The constraint 'iconicDimension3d.3' is cleaned.
Rule # 7 (LIF_3) => The constraint 'MficonsPlug.4' is cleaned.
Rule # 8 (LIF_3) => The constraint 'iconicDimension3d.4' is cleaned.
Rule # 9 (LIF_3) => The constraint 'iconicDimension3d.1' is cleaned.
Rule # 10 (LIF_3) => The constraint 'iconicDimension3d.2' is cleaned.
Rule # 11 (LIF_3) => The constraint 'iconicDimension3d.3' is cleaned.
Rule # 12 (LIF_3) => The constraint 'iconicDimension3d.4' is cleaned.
Rule # 13 (DOC_4) => Counter for links set on document.
Priority Level 3 : 13

Corrected errors : 13 / 13

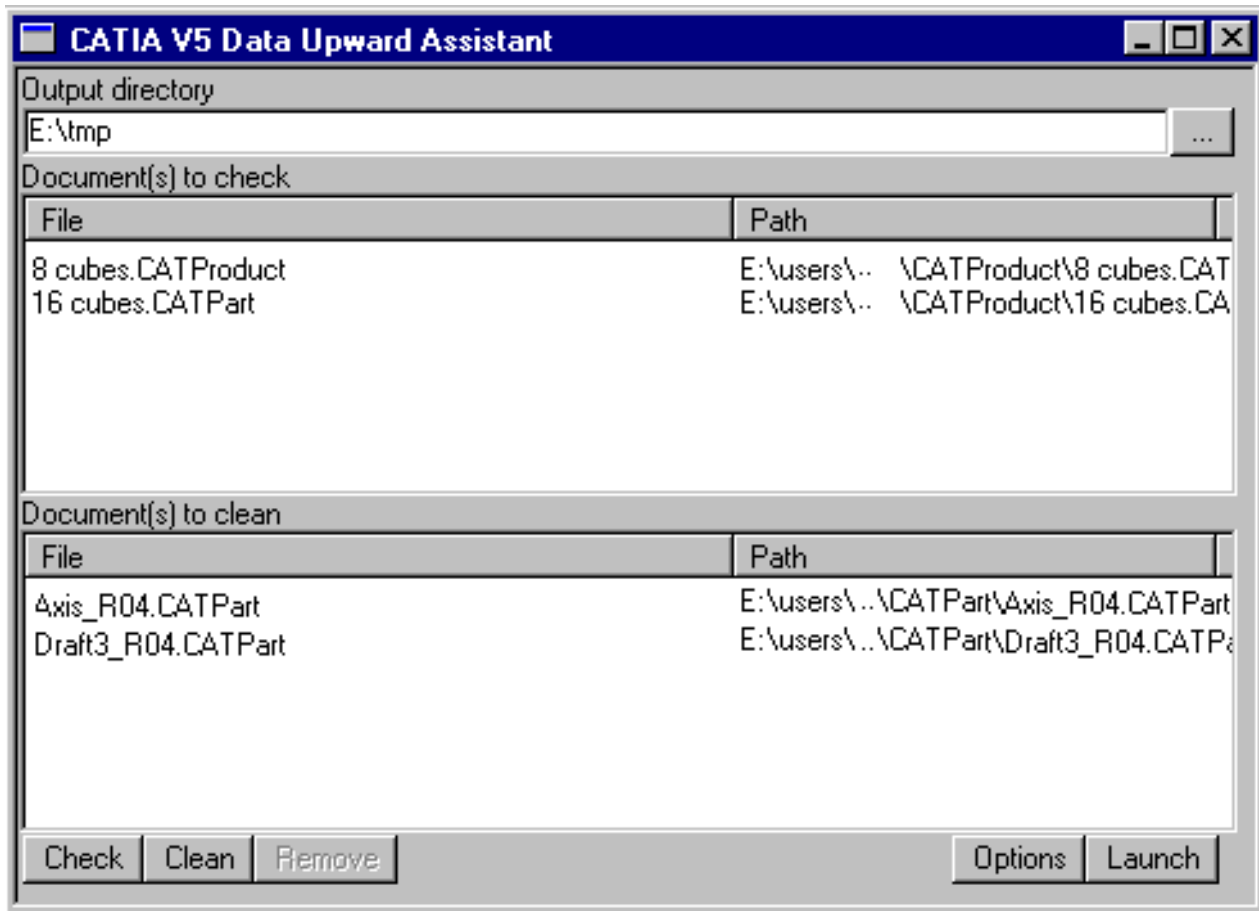
##### End of Report For Clean Process #####

-- Save Operation succeeded --
```


The status bar at the bottom of the browser window shows "Done" and "My Computer".

Example3: In Check / Upward mode on one or several CATProducts

The selection of one or several CATProducts and the launching of the Data Upward Assistant for CATProducts is exactly the same as for CATParts.



You can get the same document format (.htm) with the tables entitled **Report for Checked Documents** and **Report for Cleaned Documents**.

 Out of a CATIA session, you can have several documents in the **Report/Return Codes** field.

The upgraded documents are automatically saved in the path directory you have previously specified. In this case, the htm file reports that the upgraded documents have been saved in the **Save** field of the **Report for Cleaned Documents** table:

The .txt file (**.cleaner_traces.txt**) is available by activating the hyperlink in the htm report or by navigating in a Windows Explorer (in the same folder than the htm file).

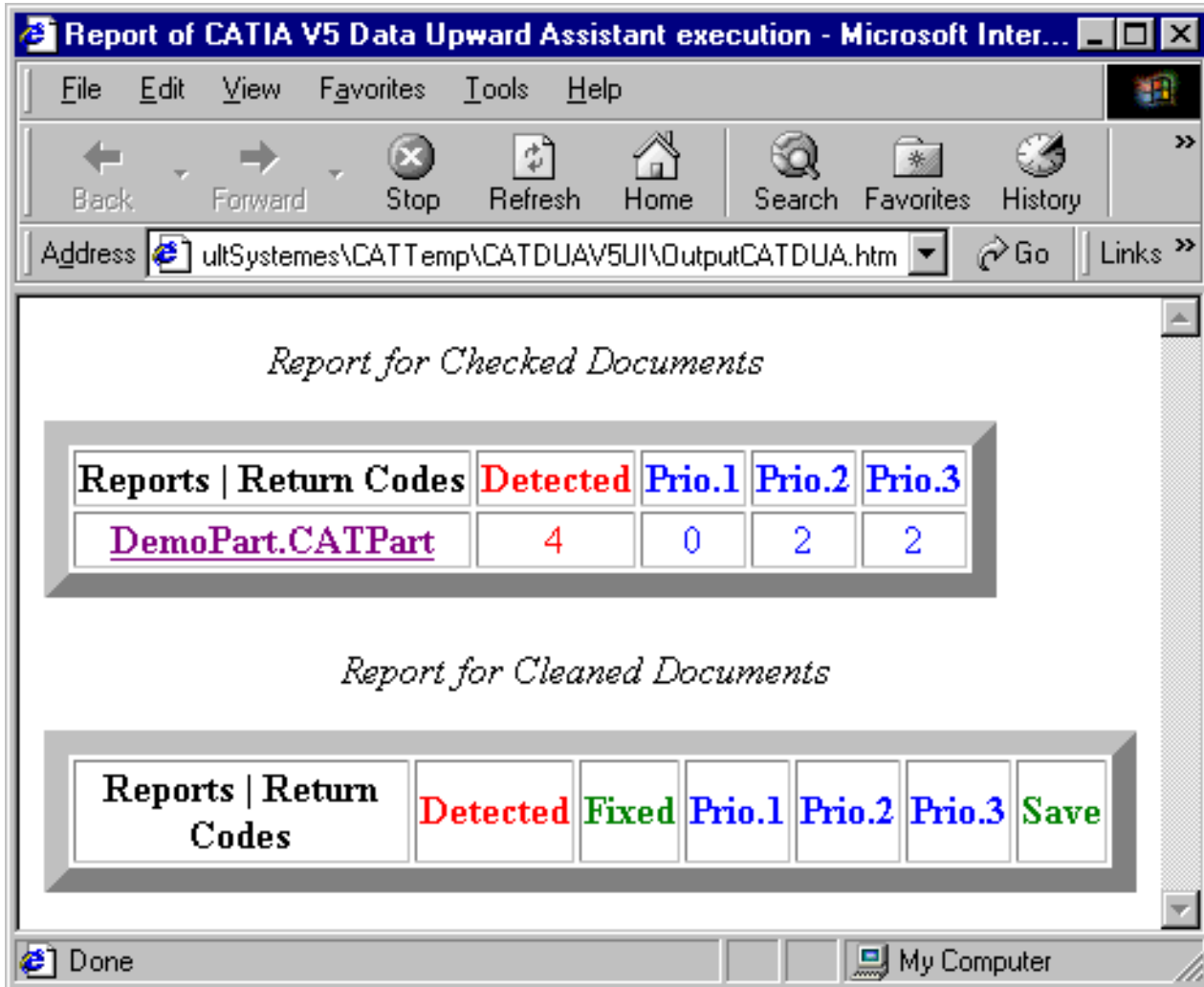
This file delivers the check results and the upward results. It informs you about the status of the return codes, the number of upgraded return codes out of the total number of return codes.

In a CATIA session

The check/clean results are stored in a htm file. This htm file is associated to a txt file which gives more details (the directory is given in the CATIA V5 Data Upward assistant Information message or in the **CATIA V5 Data Upward Assistant Information** field of the CATDuaV5 dialog box).

Example 1: In check mode on a CATPart

When you choose the check mode, the htm report only gives results of the check execution. You can find the details of those results in the txt file associated to the htm file.

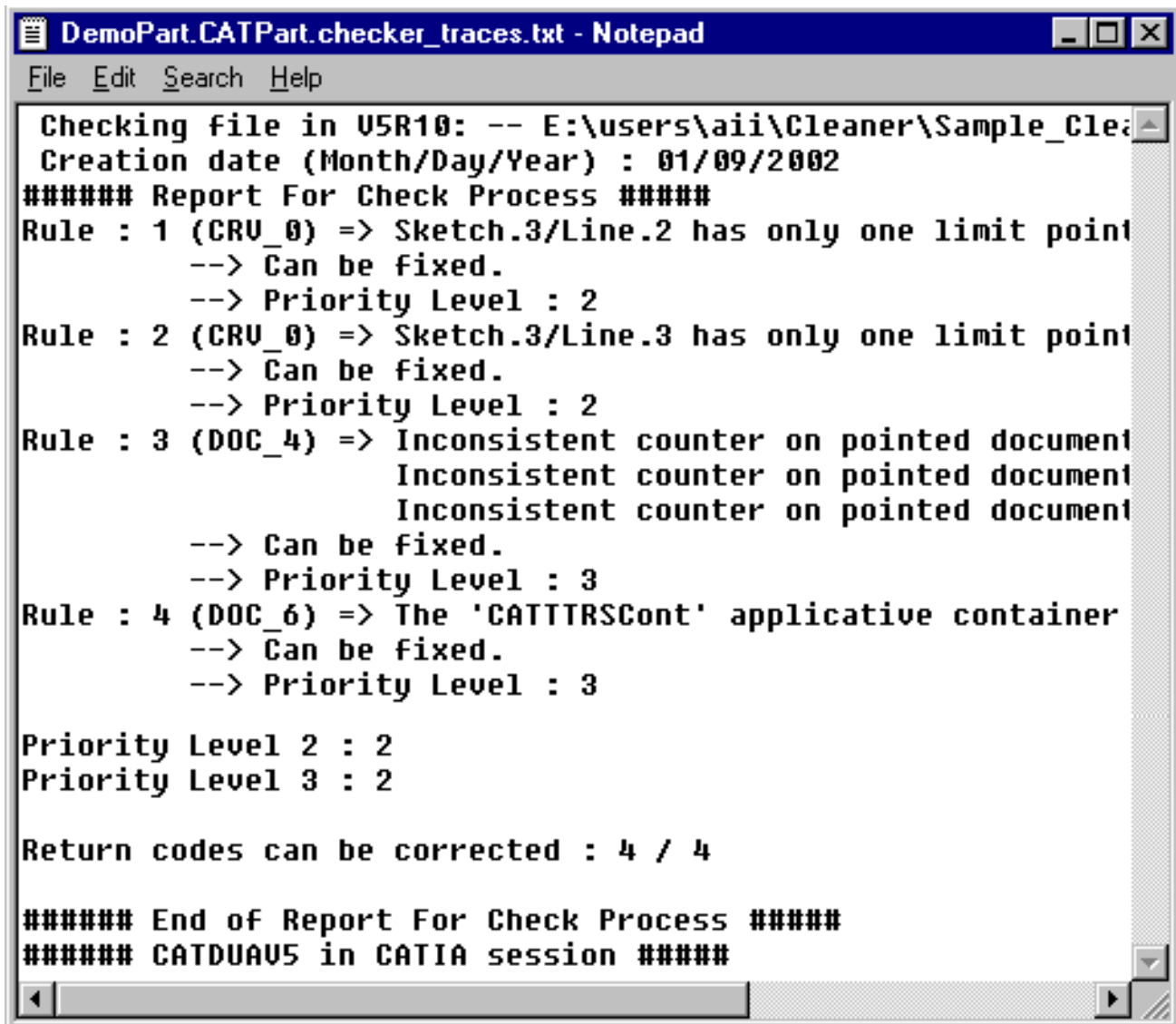


The delivered information in the htm result file is:

- name of the CATIA document you have checked
- number of the detected return codes
- number of the fixed return codes
- number of the priority1 / priority2 / priority3 return codes
- hyperlink pointing at a .txt file which gives the details of the results (in the **Reports - Return Codes** column).

The .txt file (**.checker_traces.txt**) is available by activating the hyperlink in the htm report or by navigating in a Windows Explorer (in the same folder than the htm file).

This file delivers the check results and informs you about the status of the return codes, if they can be repaired or not.

A screenshot of a Notepad window titled "DemoPart.CATPart.checker_traces.txt - Notepad". The window contains a text report with the following content:

```
Checking file in U5R10: -- E:\users\aii\Cleaner\Sample_Clea
Creation date (Month/Day/Year) : 01/09/2002
##### Report For Check Process #####
Rule : 1 (CRU_0) => Sketch.3/Line.2 has only one limit point
--> Can be fixed.
--> Priority Level : 2
Rule : 2 (CRU_0) => Sketch.3/Line.3 has only one limit point
--> Can be fixed.
--> Priority Level : 2
Rule : 3 (DOC_4) => Inconsistent counter on pointed document
                    Inconsistent counter on pointed document
                    Inconsistent counter on pointed document
--> Can be fixed.
--> Priority Level : 3
Rule : 4 (DOC_6) => The 'CATTRSCont' applicative container
--> Can be fixed.
--> Priority Level : 3

Priority Level 2 : 2
Priority Level 3 : 2

Return codes can be corrected : 4 / 4

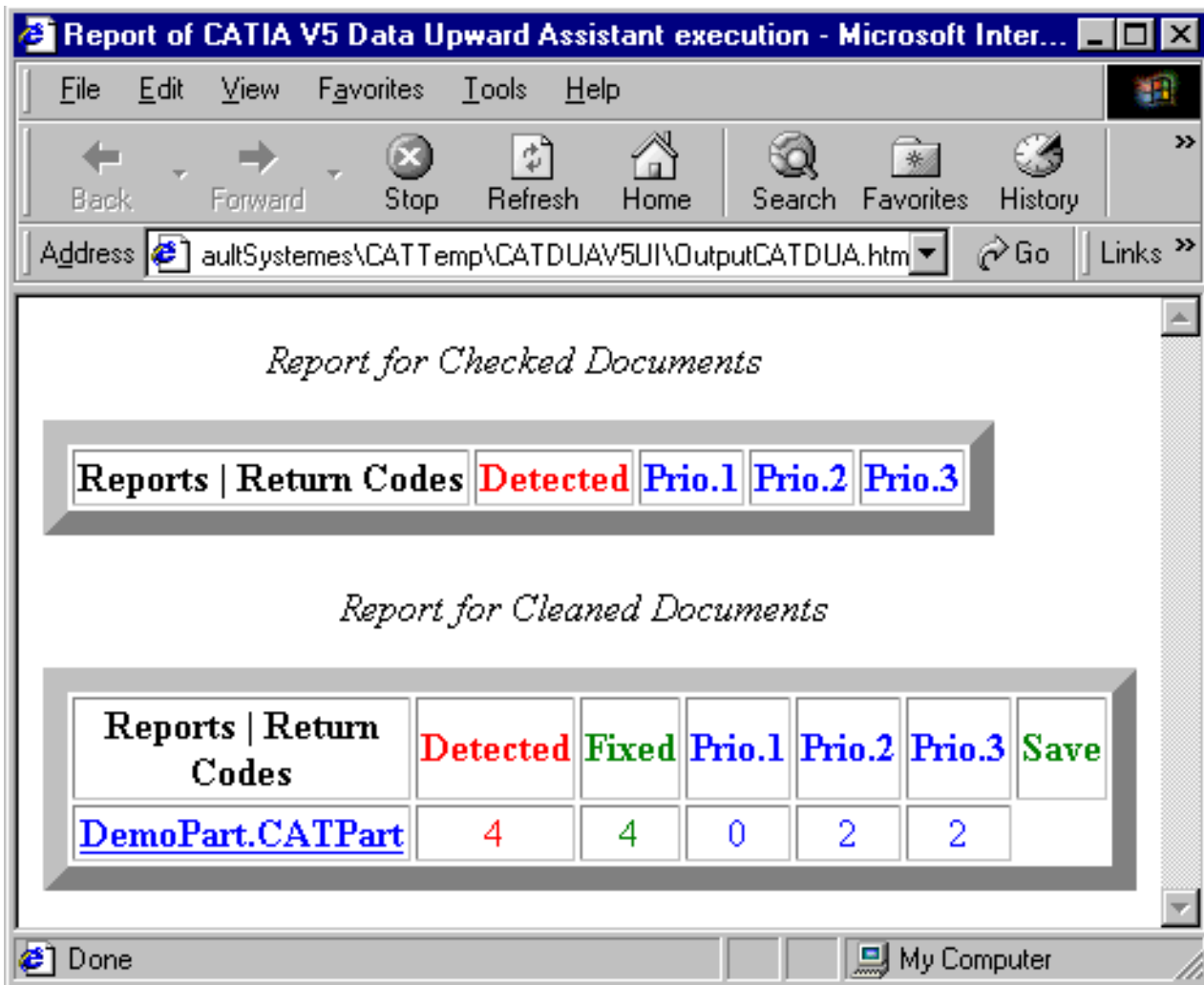
##### End of Report For Check Process #####
##### CATDUAU5 in CATIA session #####
```

You can have access to the following information:

- **Priority Level:** corruption level of the file.
- **Return codes can be corrected:** the number of return codes that can be solved in the upgrade mode.

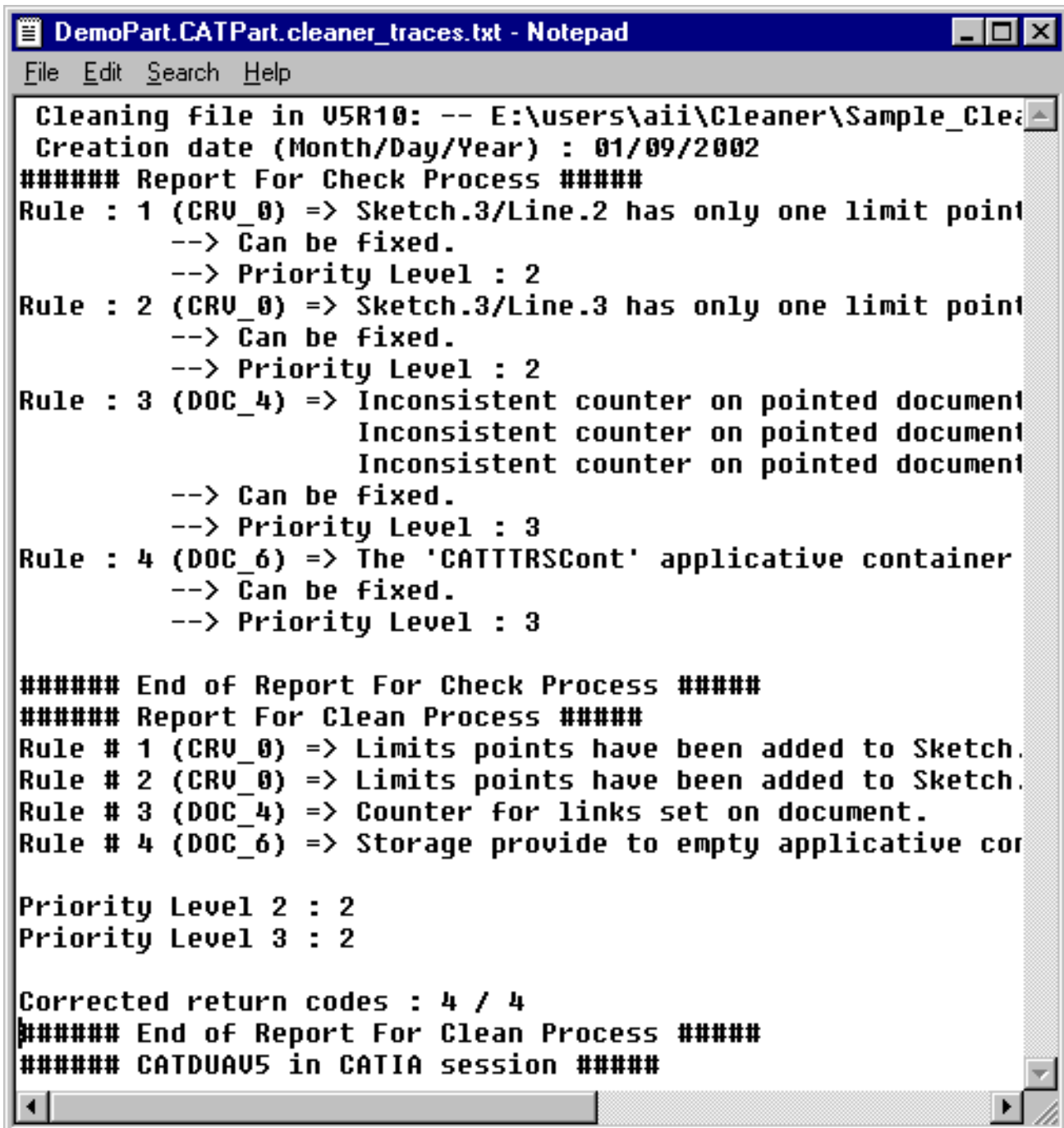
Example2: In upward mode on a CATPart

When you choose the upward mode, the htm report only gives results of the upward execution. You can find the details of those results in the txt file associated to the htm file (both the check report and the upward report).



 The upgraded document is not automatically saved, this is why the **Save** field is empty. If desired, you can save manually the upgraded document.

The .txt file (**.cleaner_traces.txt**) is available by activating the hyperlink in the htm report or by navigating in a Windows Explorer (in the same folder than the htm file). This file delivers the check results and the clean results. It informs you about the status of the return codes, the number of upgraded return codes out of the total number of return codes.



```
File Edit Search Help
Cleaning file in U5R10: -- E:\users\aii\Cleaner\Sample_Clea
Creation date (Month/Day/Year) : 01/09/2002
##### Report For Check Process #####
Rule : 1 (CRU_0) => Sketch.3/Line.2 has only one limit point
--> Can be fixed.
--> Priority Level : 2
Rule : 2 (CRU_0) => Sketch.3/Line.3 has only one limit point
--> Can be fixed.
--> Priority Level : 2
Rule : 3 (DOC_4) => Inconsistent counter on pointed document
Inconsistent counter on pointed document
Inconsistent counter on pointed document
--> Can be fixed.
--> Priority Level : 3
Rule : 4 (DOC_6) => The 'CATITRSCont' applicative container
--> Can be fixed.
--> Priority Level : 3

##### End of Report For Check Process #####
##### Report For Clean Process #####
Rule # 1 (CRU_0) => Limits points have been added to Sketch.
Rule # 2 (CRU_0) => Limits points have been added to Sketch.
Rule # 3 (DOC_4) => Counter for links set on document.
Rule # 4 (DOC_6) => Storage provide to empty applicative cor

Priority Level 2 : 2
Priority Level 3 : 2

Corrected return codes : 4 / 4
##### End of Report For Clean Process #####
##### CATDUAU5 in CATIA session #####
```

In batch mode

When you use the Data Upward Assistant in batch mode (CATDUAU5 Batch), you can find the .htm and the .txt report files in the output directory you have specified.

Return Codes Detected by the Data Upward Assistant



The Data Upward Assistant capabilities are to check structural data within a CATProduct, CATPart, CATDrawing, CATAnalysis, CATProcess, and to upgrade (modify) the data structure.

You can find here a description of several return codes detected by the Data Upward Assistant including:

- levels of severity of the detected errors
- symptoms (user view) of the detected errors
- technical problems
- CATDUA V5 operations
- results of the CATDUA V5 execution (user view)



Priority levels

Priority Level Legend:

- 1:** **Priority One Return Code:** Upgrading action may lead to data deletion.
- 2:** **Priority Two Return Code:** Upgrading action may lead to data modification (without deletion).
- 3:** **Priority Three Return Code:** Unimportant error. Upgrading action without huge impact on data.
- (*)** This symbol means that the rule is not executed on a document coming from a PDM system (ENOVIA VPM, ENOVIA LCA).

Error	Level of Severity	Domain	Symptom (User View)	Technical Problem	CATDUA Operation	After CATDUA execution (User View)
ASD_1	3	Assembly Design	No visibility (when a connection is deleted the constraint is not)	Unused elements (constraints) are present	Deletion of unused elements	Document is the same but smaller
ATT_2	3	Product Structure	No visibility (can appear when an assembly update fails)	Internal (and unnecessary) attribute incorrectly valuated	Unset the value of this attribute	No visibility
ATT_3	1	Product Structure	Send To command shows a link to a CGR in V5Cache (concern only V5R1 CATProducts)	Internal (and unnecessary) attribute incorrectly valuated	Deletion of unnecessary link	Send To command does not show this link

ATT_4	2	Product Structure	The result of an update (in terms of position) is not correct even if no error is reported	Internal value to determine whether an update (in position) is needed is incorrectly valuated	Set the internal attribute to the correct value	Update is required. After the update the part is correctly positioned
BDR_0	1	PlantShipModeler	The user will be prompt with error panel if an old version of 2D/3D modeling exist.	New 2D/3D modeling would eliminate the dependency of schematic documents, reduce data size, support ENOVIA LCA revision and configuration.	Upgrade existing, 2D/3D modeling without direct pointing to schematic/fake functions.	No more error panel and reduced model size.
BST_0	3	PlantShipModeler	None	Without the declaration of backup startup class, the document cannot be loaded without the associated CATfct	Declare object's backup startup class	None
CDC_0	1	PlantShipModeler	The user will be prompt with error panel if an invalid version of cross document connections exists.	Publication based on cross document connection would support ENOVIA LCA revision and configuration	Use publication to support cross document connections.	No more error panel
CST_1	2	Sketcher	Some sketcher constraints are not usable anymore (always broken)	Internal attribute to determine the type of the constraint is not valuated	Set the internal attribute to the correct value	Update is required. The constraint can be used again
DOC_3	2	SpecsModeler	No visibility (can lead to the lost of a link until V5R10)	Internal counter to determine whether a link to a document has to be kept not correctly valuated	Set the counter to the correct value	No visibility
DOC_4	3	SpecsModeler	Unused external links shown in Edit Links	Internal counter not correctly valuated	Set the counter to the correct value	In Edit Links, there are no more unused external links
DOC_6	3	SpecsModeler	A document cannot be stored in ENOVIA VPM	Applicative container with no stream in the document	Creation of an empty stream for this applicative container	The document can be stored in VPM
DWS_0	2	Drafting	Impossible to select a view (not allowed), Grid disappear, or impossible to activate Drafting workbench	Internal value which determines the order of the views is not set correctly	Internal value set correctly : order of the views are OK	Described problems disappear

FGM_1	3	Topological Objects	Size of CATPart abnormally huge comparatively to the number of elements	Some vertices have unnecessary geometric data (linked curves / surfaces) Some surfaces are over defined (unnecessarily extrapolated)	Remove unnecessary geometric data	Part is the same but smaller
FTA_1	1 (*)	3D Functional Tolerancing & Annotation	Sometimes, unused external links (shape representation of a product) shown in Edit Links or File Desk	The TTRS (Technologically and Topologically Related Surface) features become invalid because their StartUp has been removed from the container	Deletion of the invalid TTRS features	In Edit Links, there are no more unused external links
FTA_2	3	3D Functional Tolerancing & Annotation	No visibility	Unused elements (RGE, Reference Geometrical Elements) are present	Deletion of unused elements	Document is the same but smaller
FTA_3	3	3D Functional Tolerancing & Annotation	No visibility	Unused elements (TTRS) are present	Deletion of unused elements	Document is the same but smaller
GSD_0	3	Generative Shape Design	Sometimes, the user can only see invalid "Parent / Children" or impact analyses.	An object is aggregated under "Component" attribute but it is never used elsewhere.	Suppresses these useless objects.	Visually no impact, better 'Parent / Children' or impact analyses, reduces the size of the document.
GST_0	2	Generative Shape Design	The user opens a CATPart with an unidentified object (broken icon) called RollingOffset.	The Start-up inheritance of the object is broken, it is due to incorrect CATHybridShape.feacatalog construction.	Re-plugs the broken instance with the correct start-up in the CATHybridShape.feacatalog.	Object now corresponding to a valid Rolling Offset.
IGS_0	3	Sketcher	No visibility (when constraint deleted) (severity 3 since R9SP3)	Unused elements (phantom operators) are present	Deletion of unused elements	Update is required. Document is the same but smaller
JNT_1	1	DMU Kinematics	Mechanism cannot be simulated	One or more kinematic joint axis are corrupted	Deletion of corrupted joint	Some joints are deleted. If you want to play again the Mechanism, you have to recreate the deleted joints.

JNT_2	1	DMU Kinematics	Mechanism cannot be simulated	One or more geometries pointed by a kinematic joint are corrupted	Deletion of the corrupted joint	Some joints are deleted. If you want to play again the Mechanism, you have to recreate the deleted joints.
JNT_3	1	DMU Kinematics	This Message is displayed when trying to play a mechanism: "There is no Kinematic Data"	Several Mechanism containers exist in the same CATProduct document	Deletion of all Mechanism containers except the first non-empty one	All the previous Mechanisms are restored correctly
KWE_1	2	Knowledge	No visible symptom	Knowledge Objects not aggregated	If these objects are not pointed, they are destroyed. If it is a parameter that is pointed, it is added to the parameter set. If it is a relation that is pointed, it is added to the relation set.	No impact
KWE_2	2	Knowledge	a "!" information is displayed on a broken relation	A relation in a CATPart document points to elements in other documents.	The relation is destroyed	Broken relation disappears
KWE_3	2	Knowledge	The output value of a relation is not consistent. After forcing relation evaluation, its value changes.	The output parameter has been saved with a wrong value	The evaluation is forced	The output parameter is set to the good value.
KWE_4	2	Knowledge	A Knowledge Check is broken. "!" information is displayed	An error in the stream involves bad pointing in the Check attributes	The links are restored correctly	The check is not broken
LIF_1	1	Mechanical Modeler	Size of CATPart abnormally huge comparatively to the number of elements	Unused Elements (Geometrical feature) are present	Deletion of unused elements	Part is the same but smaller
LIF_2	3 (*)	Product Structure	Unused external links (shape representation of a product) shown in Edit Links or FileDesk	An external link is not used anymore by the CATProduct but has not been suppressed	The unnecessary external links are deleted	In Edit Links, there are no more unused external links

LIF_3	3	Mechanical Modeler	No visibility except size of the CATProduct document (constraints deletion cases)	Unused Elements (mechanical constraints) are present	Deletion of unused elements	Document is the same but smaller
LIF_4	3 (*)	Product Structure	No visibility except size of the document (when an Isolate command has failed)	Unused elements are present	Deletion of unused elements	Document is the same but smaller
MAT_1	1	Material	Some materials previously applied on geometry are missing (not visible in the tree and not applied to geometry)	Internal attribute has multiple values instead of one (multiple material container)	Deletion of unnecessary values	The missing materials are restored and are applied correctly
MFG_0	1	Manufacturing	No visibility except abnormal size of CATProcess document	Unused machining feature is not deleted (activities cases)	Delete the unused machining feature	Document is the same but smaller
MFG_1	1	Manufacturing	No visibility except abnormal size of CATProcess document	Unused pattern definition is not deleted (pattern cases)	Delete the unused pattern definition	Document is the same but smaller
MFG_2	2	Manufacturing	No visibility except abnormal size of CATProcess document	The body referenced by a manufacturing geometry in SMART NC mode is duplicated	Delete the duplicated body	Document is the same but smaller
MFG_3	1	Manufacturing	No visibility except abnormal size of CATProcess document	Toolpath not referenced by an activity is not deleted	Delete the unused toolpath	Document is the same but smaller
MGN_0	2	Mechanical Modeler	The result of an update (geometric) is not correct. No errors are reported.	Internal information has multiple values instead of one	Deletion of unnecessary values	Update is required. The result of the update is correct
MMR_1	2	Mechanical Modeler	An old CATPart cannot be used in an assembly (Insert Component fails)	The link between the shape representation and the product is broken	Recreation of the link	This CATPart can be used again

PIC_0	3	Drafting	A Drawing Picture disappears after a copy/paste	Non aggregated Picture	Deletion of non aggregated Drawing Picture	The document is smaller
SMG_9	1	Sketcher	A modification of a constraint is not taken into account by an update of the sketch	Internal value to determine if a constraint impact a sketch is not correctly valuated	Set the attribute to the correct value	The result of the update is correct
SKT_24	3	Sketcher	No visibility (severity 3 since R9SP3)	Unused elements (Brep aggregated under sketch) are present	Deletion of unused elements	Document is the same but smaller
UAV_4	3 (*)	Product Structure	Unused Links (connectors from constraints) are shown in the Send To command	Internal attribute to determine the constraint pointed by a connector is not correctly valuated	Set the attribute to the correct value	Those links are not visible anymore
URL_1	3	Knowledge	Bad performance when opening a document	An URL points a document through an invalid attribute type	Attribute is turned into the correct type	Document is opened with correct performance



To have the complete list of the return codes detected by the Data Upward Assistant, please refer to [List of the Detected Return Codes](#).



List of the Detected Return Codes



The Data Upward Assistant capabilities are to check structural data within a CATProduct, CATPart, CATDrawing, CATAnalysis, CATProcess, and to upward (modify) the data structure.

You can find here the list of all the return codes that can be detected by the Data Upward Assistant and a description.

The Data Upward Assistant is useful in the following situations:

- before recovering external data
- before going into a new CATIA release
- broken links when opening CATProducts
- incidents when updating a component (for instance, Sketch update)
- the Edit-Links panel appears: some documents are found but they have no references.
- performance problems when opening a CATProduct (because some elements have lost their links).



Priority levels

Priority Level Legend:

(1):

Priority One Return Code: *Upgrading action may lead to data deletion.*

(2):

Priority Two Return Code: *Upgrading action may lead to data modification (without deletion).*

(3):

Priority Three Return Code: *Unimportant error. Upgrading action without huge impact on data.*

(*)

This symbol means that the rule is not executed on a document coming from a PDM system (ENOVIA VPM, ENOVIA LCA).

List of the detected return codes

1. ObjectSpecsModeler:

DOC: linked document (container root)

DOC_1 (2) (*): non typed document in a link => Cleaner: Document typed in the link.

DOC_3 (2): missing links => Cleaner: Link counter set up-to-date.

DOC_4 (3): ghost links=> Cleaner: Link counter set up-to-date.

DOC_5 (2): destroyed links => Cleaner: Re-build the missing link.

DOC_6 (3): applicative containers are lost => Cleaner: Empty stream added.

CAT: Catalog

CAT_0 (2) (*): feature catalog with no name => Cleaner: Deletion of the link to this catalog.

2. Product Structure:

BRK: Broken Object

BRK_0 (1) (*): broken objects not aggregated => Cleaner: Deletion of objects.

LIF: Product Lifecycle

LIF_2 (3) (*): objects not reachable starting from root-product => Cleaner: Deletion of objects.

LIF_4 (3) (*): non-destroyed objects in CATProdCont container => Cleaner: Deletion of objects.

LIF_7 (3) (*): non-destroyed connections in CATProdCont container => Cleaner: Deletion of objects.

LIF_8 (3) (*): non-destroyed connectors in CATProdCont container => Cleaner: Deletion of objects.

LIF_9 (3): non-destroyed product bag reps in CATProdCont container => Cleaner: Deletion of objects.

ATT: Rule Attribute

ATT_2 (3): unused attribute _UpdateError => Cleaner: Deletion of attribute.

ATT_3 (1): unused attribute (activrep) valuated => Cleaner: Unset of the attributes.

ATT_4 (2): update stamp incorrect on position attribute => Cleaner: Update of UpdateStamp.

UAV: Product structure - Unexpected Attribute Value

UAV_0 (2): illegal position attribute => Cleaner: Position is set to identity.

UAV_1 (1): incorrect overloading for position attribute => Cleaner: Switch in flexible products.

UAV_2 (1): component's list with blank => Cleaner: Deletion of blanks.

UAV_4 (3) (*): invalid object pointed by a connector => Cleaner: Link set correctly.

UAV_6 (3): invalid connector pointed by a connection => Cleaner: Connection synchronized.

UAV_7 (2): instance of Product with overloaded contextuality => Cleaner: Product instance synchronized.

UAV_8 (2): a Product with many representations (no Visualization when opening) => Cleaner: Product instance synchronized.

UAV_9 (3): connector with incorrect connections => Cleaner: incorrect connections deleted.

UPG: Upgrade

UPG_0 (3): invalid multi-representation (V5R2 model) => Cleaner: Upgrade of multi-representation.

DOC: Linked Document (container root)

DOC_7 (2) (*): no save link to a sub-product => Cleaner: Link switch as a save-link.

SYN: Synchronization

SYN_1 (3): Multi-Instanciation of one reference by synchronization. => Cleaner: deletion of the feature.

AQT: Quality

AQT_1 (3): the quality on Publication instance is different from the Publication startup. => Cleaner: change quality from neutral to in.

3. Mechanical Modeler

LIF: Product Lifecycle

LIF_1 (1): The size of the CATPart is abnormally huge comparatively to the number of elements and unused elements (Geometrical feature) are present => Cleaner: Deletion of unused elements, the Part is the same but smaller.

LIF_3 (3): No visibility except the size of the CATProduct document (constraints deletion cases) => Deletion of unused elements; the document is the same but smaller.

LIF_5 (3): non-referenced elements in the container CATSelSetsCont => Cleaner: Deletion of elements.

LIF_10 (3): Not pointed or useless proxy objects => Cleaner: Deletion of the useless proxy objects.

LIF_11 (3): Internal object unused in applicative container => Cleaner: Deletion of the useless objects.

MMR: Mechanical Modeler

MMR_0 (1): corruption of the 3 reference planes => Cleaner: Restore valid reference planes.

MMR_1 (2): ShapeRep not pointed by a Product => Cleaner: Reset link.

MMR_2 (2): No Maintool in the Part => Cleaner: Creation of a default Maintool.

MGN: Mechanical Generic Naming (topology)

MGN_0 (2): more than one topological result => Cleaner: Deletion of over results.

MTR: Mechanical Tools

MTR_0 (2): tool's result not properly plugged => Cleaner: Result plugged.

MTR_1 (3) (*): Body's reference is lost => Cleaner: Reference is set to Start-Up.

MTR_2 (2): Shape Features not properly plugged => Cleaner: Re-plug of the Shape Feature.

MTR_3 (3): invalid valuation of VisuOnOff attribute => Cleaner: Reset attribute.

MFT: Shape Features Tools

MFT_0 (2): active/inactive status mismatched regardless to attribute structure => Cleaner: Reset attribute.

MFT_1 (1): inactive Feature storing the result of the previous feature => Cleaner: No more result in the feature.

MFT_2 (1): result of a shape feature valuated with result of the previous active shape feature => Cleaner: Result regenerated.

MFT_3 (1): inactive Shape Feature does not have its attribute saved or it is unset => Cleaner: Attribute regenerated and/or re-evaluated.

GST: Generative Surface Tool

GST_0 (2): The user opens a CATPart with an unidentified object (broken icon) called RollingOffset; startup lost for a GSMRollingOffset => Cleaner: Re-plug the startup.

GSD: Generative Shape Design

GSD_0 (3): unused aggregated feature => Cleaner: Deletion of these useless features.

GSD_1 (1): reference planes are set as datum or independent spec => Cleaner: Correction of the Visu Graph attribute, aggregation of the moved reference plane in current OpenBody and creation of a new reference plane.

CTX: Context

CTX_1 (3) (*): multi-contextuality for a Part => Cleaner: Deletion of over contexts.

CST: Constraints

CST_1 (2): non-typed constraint => Cleaner: The constraint is typed.

CST_2 (2): attributes quality mismatched => Cleaner: Quality modification.

CST_3 (1): not valuated mandatory attributes on constraints => Cleaner: Deletion of constraints.

UDF: User Feature

UDF_0 (2) (*): user-feature with an invalid input => Cleaner: Re-plug of inputs.

4. Assembly:

ASD: Assembly Design (constraints)

ASD_1 (3): isolated constraint => Cleaner: Deletion of constraints.

ASD_2 (3): mismatched storage of constraint => Cleaner: Deletion of constraints.

ASD_3 (3): constraint with lost data => Cleaner: Synchronization of the constraints.

ASD_4 (3): attributes quality mismatched => Cleaner: Modification of the quality.

AFI: Assembly Feature

AFI_1 (3): assembly feature pointing wrong inputs => Cleaner: Correction of wrong links "assembly feature->input".

5. Analysis:

SAF : Features (Spec view)

SAF_2 (2): feature has no explicit image => Cleaner: Creation of an explicit image.

SAF_3 (1): feature contains invalid explicit data => Cleaner: Deletion of this data.

SAF_5 (2): nodes with wrong activity status => Cleaner: Correction of the status.

6. Sketcher:

SKT: Sketcher

SKT_0 (1): sketch with no father => Cleaner: Deletion of Sketch.

SKT_24 (3): Fsur of the _FtrList must be referenced => Cleaner: Deletion of the Fsur.

SKT_25 (3): mismatched version on sketch => Cleaner: Change version.

SMG: SolveManager (Sketch object)

SMG_4 (2): ImportedGeom attribute size must be the same as the number of operators => Cleaner: Resize list.

SMG_5 (2): ImportedGeom attribute with no operator => Cleaner: Reevaluation of attribute.

SMG_6 (2): mismatched ImportedGeometry attribute => Cleaner: Devaluation of attribute.

SMG_9 (2): invalid number of constraints => Cleaner: Re-evaluation of attribute.

SMG_16 (2): non-impacted Solve Manager at update => Cleaner: Set constraint attribute to neutral.

SMG_17 (2): invalid OutputGeoms attribute => Cleaner: Revaluation of attribute.

IGS: Imported GeomSet

IGS_0 (3): invalid phantom operator => Cleaner: Deletion of the operator

SKS: Sketch Support

SKS_1 (2): Update cycle not detected => Cleaner: Change structure of feature to detect cycle.

CRV: Curve

CRV_0 (2): end points not found on the curve => Cleaner: Add limit points.

7. Material:

MAT: material

MAT_1 (1): more than one material container in a Part => Cleaner: Deletion of over containers.

MAT_2 (3): orphans elements in the material container => Cleaner: Deletion of orphans.

MAT_3 (2): wrong visualization of a material => Cleaner: Reapply of material. Note that this error could sometimes be impossible to repair if the material can not be retrieved (for example, if the material is located in an inaccessible material catalog).

MAT_4 (3): invalid material link => Cleaner: Deletion of link.

8. Drafting:

DRW: Drawing

DRW_0 (1) (*): more than one drawing in the document => Cleaner: Deletion of unused drawings.

DRW_1 (1): corrupted sheet => Cleaner: Deletion of the sheet.

DWS: Drawing Sheet

DWS_0 (2): views not well organized in a sheet => Cleaner: Re-ordinate views.

DWS_1 (1) (*): corrupted sheet => Cleaner: Deletion of sheet.

DWV: Drawing View

DWS_0 (2) (*): corrupted view (not well aggregated) => Cleaner: Deletion of the view.

DVM: Drawing View MakeUp

DVM_0 (2): unused View MakeUp => Cleaner: Deletion of View MakeUp.

GIE: Generated Item

GIE_0 (2): unused GenItem => Cleaner: Deletion of GenItem.

DAF: Drawing Area Fill

DAF_0 (3): Area Fill without profile => Cleaner: Deletion of area.

DAF_1 (3) (*): Area Fill not aggregated => Cleaner: Deletion of area.

DAF_2 (3) (*): Area Fill with a missing curve => Cleaner: Deletion of area.

DAF_3 (3): Area Fill without pattern => Cleaner: Apply default pattern.

STD: Standards

STD_0 (2): too many standards in the document => Cleaner: Deletion of unused standards.

STD_1 (2): Standard not synchronized with the active sheet => Cleaner: Re-synchronization of standards.

DET: Drawing Detail (2D component)

DET_0 (3) (*): 2D Component not aggregated => Cleaner: Deletion of component.

DET_1 (3) (*): 2D Component not typed through interface => Cleaner: Deletion of component.

OLE: OLE Object

OLE_0 (2) (*): OLE object pointing at an invalid drawing => Cleaner: the OLE object is replaced by a new one.

DWG: GenDim

DWG_0 : object managing dimension generation no more used => Cleaner: Deletion of GenDim.

DWG_1 (3): object managing dimension generation not aggregated => Cleaner: Deletion of GenDim.

DWC: Callout

DWC_1 (3): Callout not aggregated => Cleaner: Deletion of Callout.

DWA: Projected Axis

DWA_1 (3): object manufacturing cut-views not aggregated => Cleaner: Deletion of Projected Axis.

DWH: Generative pattern

DWH_1 (3): generative pattern not aggregated => Cleaner: Deletion of Generative pattern.

DWP: Pattern Mapping Table

DWP_0 (3): pointer for pattern on 3D no more used => Cleaner: Deletion of object.

DWP_1 (1): pointer for pattern on 3D not aggregated => Cleaner: Deletion of object.

DAO:

DAO_0 (1) (*): Associativity pointing an external link with the document => Cleaner: Deletion of Associativity.

PIC: Drawing Picture

PIC_0 (3): Non aggregated Picture => Cleaner: Deletion of the picture.

9. Annotation:

DST: Simple Text

DST_0 (1) (*): invalid simple text => Cleaner: Deletion of object.

DCR: Connector

DCR_0 (1): problem on connector => Cleaner: Retrieve section specification in related view.

DCR_1 (3) (*): obsolete internal element for associativity => Cleaner: Deletion of object.

DDI: Dimension

DDI_0 (1) (*): invalid dimension pointing data => Cleaner: Deletion of the dimension.

DDI_1 (3) (*): dimensions pointing in another document => Cleaner: Deletion of the dimension.

DAC: Drawing Area Quotation

DAC_0 (1) (*): invalid dimension line => Cleaner: Deletion of the dimension.

DCH:

DCH_0 (3): Invalid 3D-Connector referred in constraint => Cleaner: Deletion of the constraint.

DCH_1 (2): Invalid elements on a constraint => Cleaner: Constraint is fixed.

10. 3D Functional Tolerancing & Annotation:

FTA: Functional Tolerancing & Annotation

FTA_1 (1) (*): Some features in TTRS (Technologically and Topologically Related Surface) container are not valid; the TTRS features become invalid because their StartUp has been removed from the container => Cleaner: Deletion of the invalid TTRS features.

FTA_2 (3): The RGE (Reference Geometrical Elements) feature is not used by any 3D annotations; unused elements (RGE) are present => Cleaner: Deletion of the unused RGE features.

FTA_3 (3): The TTRS feature is not used by any 3D annotations; unused elements (TTRS) are present => Cleaner: Deletion of the unused TTRS features.

11. Interactive Drafting:

DCH:

DCH_1 (2): Invalid elements on a constraint => Cleaner: The constraint is fixed.

12. SheetMetal:

UPD: Update

UPD_1 (2): invalid update stamp => Cleaner: Set update to false.

13. Manufacturing:

MFG: Manufacturing

MFG_0 (1): Machining Feature not referenced by an Activity; unused machining feature is not deleted (activities cases) => Cleaner: Deletion of the unused Machining Feature.

MFG_1 (1): Pattern Definition not referenced by a Pattern Usage; unused pattern definition is not deleted (pattern cases) => Cleaner: Deletion of the unused Pattern Definition.

MFG_2 (2): Manufacturing Geometry referencing a duplicated body; the body referenced by a manufacturing geometry in SMART NC mode is duplicated => Cleaner: Deletion of duplicated body.

MFG_3 (1): Toolpath not referenced by an Activity => Cleaner: Deletion of the unused Toolpath.

14. Knowledge:

KWE: Knowledge

KWE_1 (2): knowledge object not aggregated; bad performance when opening a document and An URL points a document through an invalid attribute type => Cleaner: Deletion of the Attribute. If the parameter is pointed by a relation, add object in parameter set; if the object is a relation, add it in the relation set).

KWE_2 (2): Knowledge Relation in a CATPrctCont points to other documents => Cleaner: The relation is destroyed.

KWE_3 (2): The output value of a relation is not consistent. After forcing relation evaluation, its value changes => Cleaner: The evaluation is forced.

KWE_4 (2): A Knowledge Check is broken. "!" information is displayed and an error in the stream involves bad pointing in the Check attributes => Cleaner: The links are restored correctly.

URL_1 (3): Bad performance when opening a document and an URL points a document through an invalid attribute type => Attribute is turned into the correct type.

15. Kinematics:

JNT: Joint

JNT_1 (1): One or more kinematic joint axis are corrupted => Cleaner: Deletion of corrupted joint.

JNT_2 (1): One or more geometries pointed by a kinematic joint are corrupted => Cleaner: Deletion of the corrupted joint.

JNT_3 (1): Several Mechanism containers exist in the same CATProduct document => Cleaner: Deletion of all Mechanism containers except the first non-empty one.

16. Generic Naming:

GNL: Generic Naming Link

GNL_1 (3) (*): BRep feature is pointing at a ghost document => Cleaner: Deletion of link.

GNL_3 (3) (*): BrpRsur, BrpWire, ... are not aggregated => Cleaner: Deletion of BrpRsur, BrpWire,...

17. New Topological Objects:

FGM: Fat Geometry

FGM_1 (3): Topological Objects; the size of the CATPart is abnormally huge comparatively to the number of elements. Some vertices have unnecessary geometric data (linked curves / surfaces and some surfaces are over defined (unnecessarily extrapolated) -> Cleaner: Removing of unnecessary geometric data, the Part is the same but smaller.



If you update the CATPart, this error may re-appear again (after the cleaning rule).

18. SpaceAnalysis :

LIF: Product Lifecycle

LIF_6 (1) (*): Dimension which are CATSpecObject_Broken => Cleaner: Deletion of the dimension and measure.

19. Camera:

CAM: Camera

CAM_1 (2): Multiples Camera Container => Cleaner: Deletion of the Container.

CAM_2 (2): Camera SpecObject_Broken => Cleaner: Creation of new Cameras and deletion of the SpecObject Broken.

CAM_3 (1): Mismatch between a SpecObject Attribute and the knowledge ones. => Cleaner: The value between the two types of attributes are compared and correctly set.

20. Part Design:

DRF: Draft

DRF_0 (3): Faces to draft, fix aggregation of Brep Features. => Cleaner: remove Rsur Features from CGMBody to PartBody.

DRF_1 (3): The default pulling direction is not compatible in pull dir object and vector. => Cleaner: The default pulling direction in object and vector is reset.

21. PlantShipModeler:

BST: Backup StartUp

BST_0 (3): The Backup StartUp is not declared on Psp Product => Cleaner: The Backup StartUp is declared.

BDR:

BDR_0 (1): Invalid version of 2D/3D modeling => Cleaner: Upgrade will eliminate direct pointing to schematic functions.

CDC: Cross Document Connection

CDC_0 (1): The user will be prompt with error panel if an invalid version of cross document connections exists => Cleaner: Use publication to support cross document connections.