White Paper

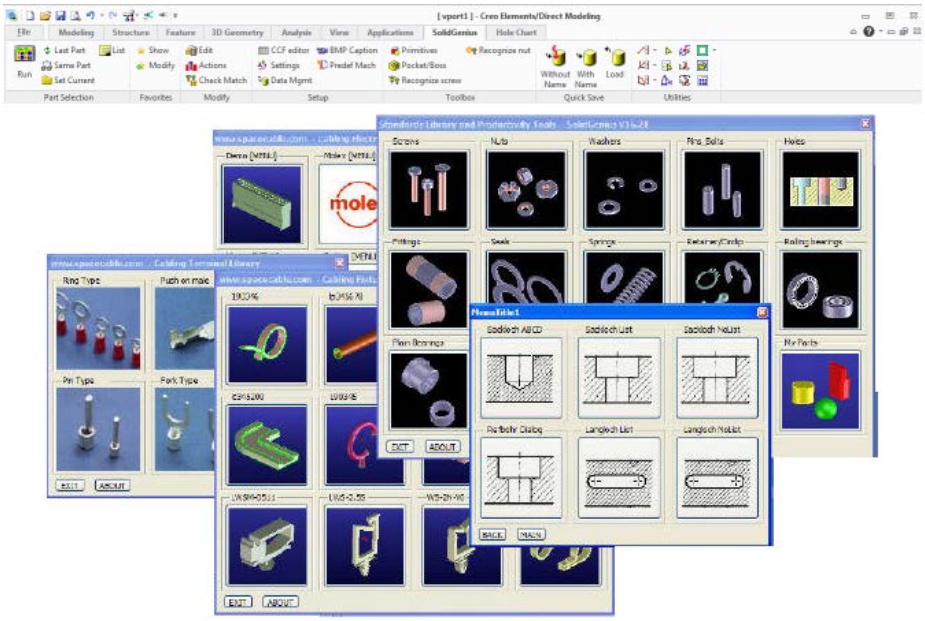
SpaceLibrary

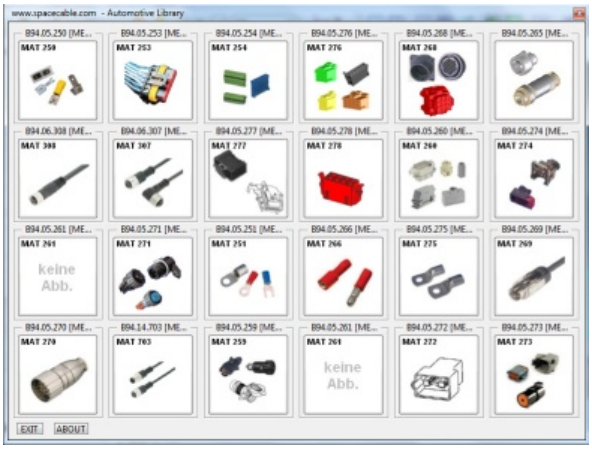
Library Management Module for Creo Elements/Direct

What is SpaceLibrary ?

SpaceLibrary is a powerful ‘Library Management Module’ to create advanced graphical user-defined menus in Creo Elements Direct without the need of any programming.

Examples of menus created by SpaceLibrary:





The benefits of SpaceLibrary

- No need to write any complex LISP programs – a menu is easily defined by a simple text file (CCF- File) created by the user. All the rest is done automatically.
- You can create as many menus as you want and run them even parallel to each other
- Supports different menu sizes and almost unlimited number of sub-menus and menu fields
- You can modify existing menus or start your own menus from scratch
- If you don't want to create menu definitions manually - additional library tools for automatic CCF creation are available from Mip
- Multi functional fields satisfy almost every wish. Different functions for each menu field (load pkg-file, call a function, open PDF , Word or Excel documents, load from ModelManager, table-driven loading, execute a Windows programme ..etc) . You can also mix fields with different functions within one single menu
- You can place your menu definition files in a central location to be used by all users or individual users can make their own menus (follows Creo customization rules)
- It is easy to use and very quick (to create a menu with SpaceLibrary takes typically under 30 minutes compared to several days if it is done by writing a LISP program)
- Open system – we can add further enhancements as customer needs arise (for example 'family of parts' was originally a customer request)

What is a CCF File ?

CCF (Catalog Creation File) is a structured text file which easily but fully describes a SpaceLibrary Menu.
It is used by SpaceLibrary Module to create customer menus for Creo.

Who can create a CCF File ?

CCF-Files can be created by anyone with a simple text editor (e.g Notepad).

I have many different parts. Should I create one big menu with many sub menus ?

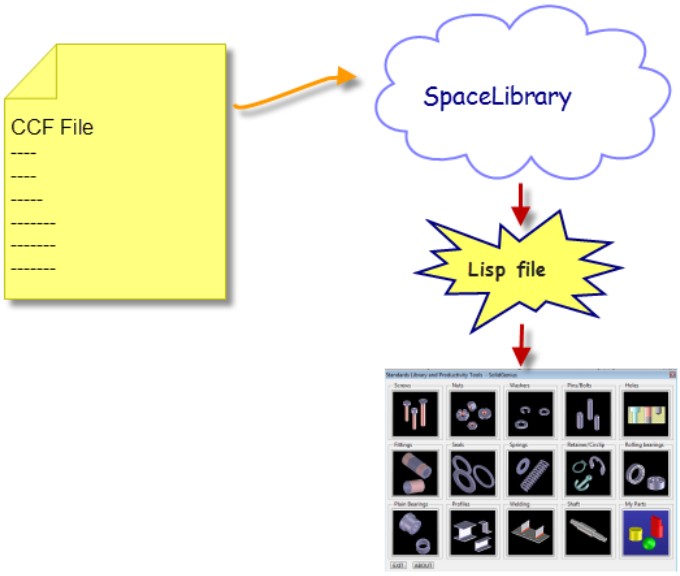
This is up to you. You can create a big menu with many submenus or you can create as many different CCF files as you wish – each of them will create a different menu - for example one menu for electrical parts, another one for screws ..etc.

Is there a limit for levels of sub menus ?

No – theoretically there is no limit. You can create several sub - sub – sub .. menus. But in practice there can be a limit due to stacking limit of the built-in LISP interpreter in Creo.

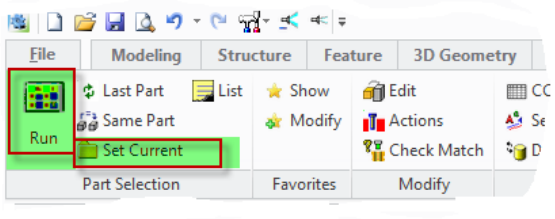
How does SpaceLibrary work ?

SpaceLibrary reads the content of CCF-File and generates automatically an intermediate LISP file. This LISP file is executed automatically in the background and it creates a graphical menu with a unique name inside Creo. This menu is shown later automatically on the screen.



How do I start SpaceLibrary ?

You don't need to start SpaceLibrary.
SpaceLibrary functions are already embedded in two SolidGenius commands 'Run' and 'Set Current'



What does [Run] function do ?

The last SpaceLibrary menu in memory– which was generated by SpaceCable – will be automatically visible on the screen.
If there has not been any menu generation yet , the default CCF-File will be taken and the menu will be generated from it. (This happens at every new start of Creo Modeling)
The default CCF File is normally the standard SolidGenius Menu.

Can I set another CCF file as default ?

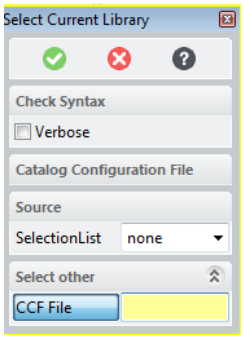
Yes. The variable 'SG_DEFAULT_CCF_FILE' defines the default CCF file.

You can modify the default CCF-File in the 'sg_start.lsp' file.

```
; Default CCF file - depending on current language setting
;(setf SG_DEFAULT_CCF_FILE (format nil "~a~a" sg_path "/My_SolidGenius_big.ccf"))

(cond

        ( (eq (sd-inq-current-language) :english)
          (setf SG_DEFAULT_CCF_FILE (format nil "~a/message/English/My_SolidGenius_big.ccf" sg_path))
          (setf SG_LANG "e")
        )

        ( (eq (sd-inq-current-language) :german)
          (setf SG_DEFAULT_CCF_FILE (format nil "~a/message/German/My_SolidGenius_big.ccf" sg_path))
          (setf SG_LANG "g")
        )
```

What does [Set Current] function do ?

Inside this dialog, you can select a CCF File and SpaceLibrary will process and create the menu



From this point on, this menu will be the 'last menu' in memory and it will be automatically shown when user hits the 'Run' button.

If I have generated different menus with the [Set current] command, all those menus are already in the memory. Can I select which menu should be shown by the [Run] command ?

No.

The 'RUN' command is written in this way, that the last menu in memory will be shown automatically.

If you want to show another menu, you should again use the [Set current] function to generate it and this will be the 'last menu' itself.

(Note: Advanced users have naturally possibility to use LISP commands to show easily any menu which is already in memory with a single command :

(oli::sd-show-dialog-shell menu_name)

Menu_name = Unique menu name as defined in CCF_File + "_MAIN"

For example:
In the below CCF File, the unique menu name is set to 'SGLIBCON2'

```
# F   ... calls a function
# P   ... calls an external program,
# LOS ... loads a pkg file from operating system,
# LWM ... loads a pkg file from WM
# LMM ... loads a pkg file from MM
#
# Please do not use prefix 'SC_' for your own pkg files !
#
SGLIBCON2 # this is a unique library name - not visible for us
SolidGenius # 1. text in the [About] box
Version 15.00  # 2.text
www.mip-group.com
All rights reserved
Fitting
120
130
 10
 10
 11
```

Once the menu is generated and it is in the memory, it can be called in the Creo command line with the following command :

(oli::sd-show-dialog-shell "SGLIBCON2_MAIN")

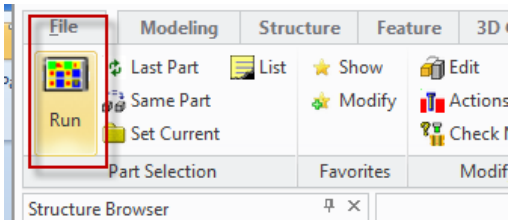
If you need further information – please contact MIP)

Do I have to generate my menu each time when I want to show it on screen ?

Each time you start Creo, SpaceLibrary menus have not been generated yet. The memory is empty. In this case SpaceLibrary must first generate the LISP file from the CCF File and execute it to generate the menu.

But – once this is done, until you exit from Creo , the menus are kept in the memory of Creo. When you want to see the menu again, you simply call the menu from the memory and no menu generation is needed – and the menu will pop up immediately on the screen.

For this reason the first call to the 'RUN' command always takes longer than next calls.

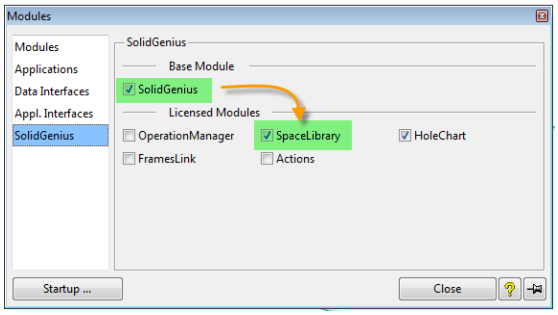


When you first call the 'Run' command, the menu is generated.
In next calls to the 'Run' command, because the menu is already there, it is quickly made visible.

Do I have to buy a license of the SpaceLibrary module to make my own menus ?

No. 'SpaceLibrary' is automatically included in every license of 'SolidGenius', SpaceCable and SpacePipe.

When you start SolidGenius, the SpaceLibrary module is also automatically started.

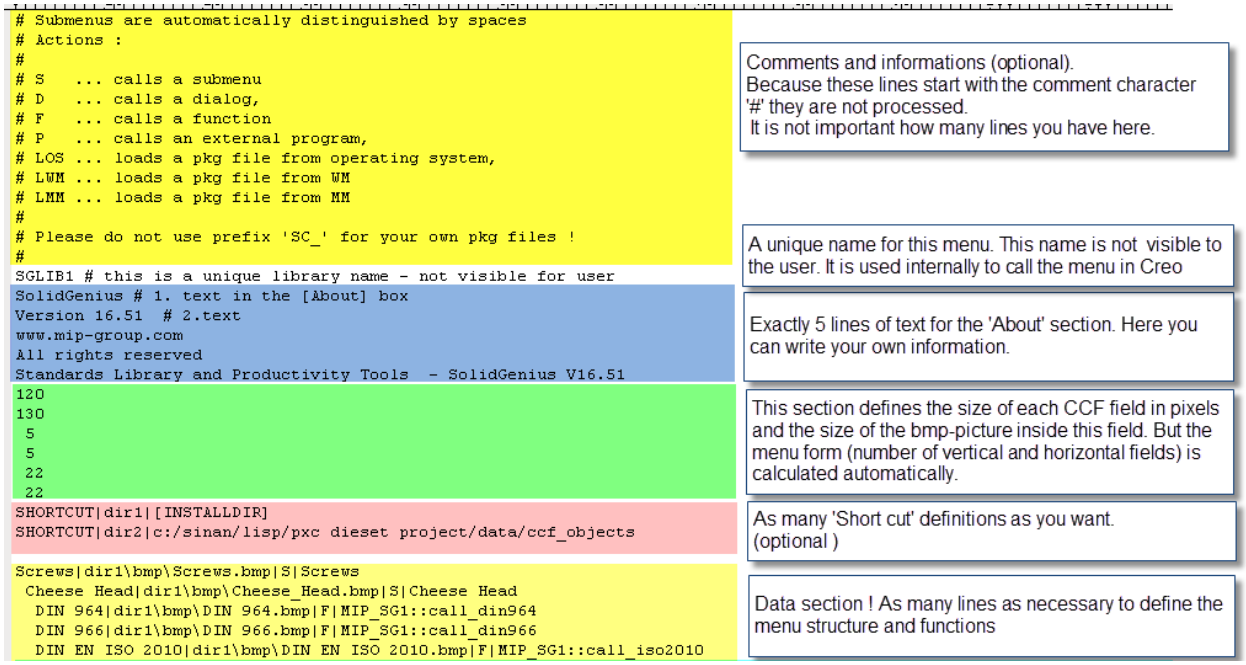


How is the CCF File structured ?

In CCF files the character '#' marks the beginning of a comment. (All text behind the '#' character will not be read by SpaceLibrary while processing the file)

Every CCF File has 6 different sections.

Two of them are optional to have.

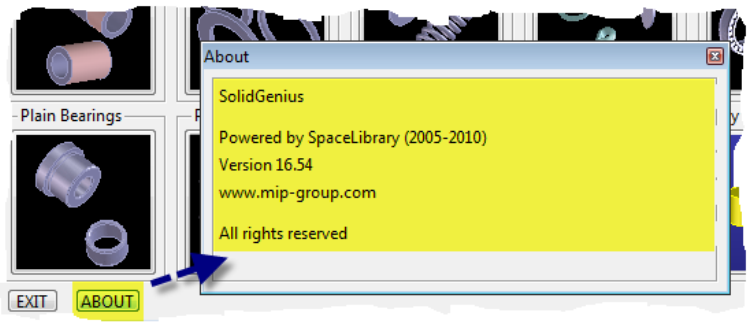



```
# Submenus are automatically distinguished by spaces
# Actions :
#
# S    ... calls a submenu
# D    ... calls a dialog,
# F    ... calls a function
# P    ... calls an external program,
# LOS ... loads a pkg file from operating system,
# LWM ... loads a pkg file from WM
# LMM ... loads a pkg file from MM
#
# Please do not use prefix 'SC_' for your own pkg files !
#
SGLIB1 # this is a unique library name - not visible for user
SolidGenius # 1. text in the [About] box
Version 16.51  # 2.text
www.mip-group.com
All rights reserved
Standards Library and Productivity Tools  - SolidGenius V16.51
120
130
 5
 5
 22
 22
SHORTCUT|dir1|[INSTALLDIR]
SHORTCUT|dir2|c:/sinan/lisp/pxc dieset project/data/ccf_objects

Screws|dir1\bmp\Screws.bmp|S|Screws
 Cheese Head|dir1\bmp\Cheese_Head.bmp|S|Cheese Head
  DIN 964|dir1\bmp\DIN 964.bmp|F|MIP_SG1::call_din964
  DIN 966|dir1\bmp\DIN 966.bmp|F|MIP_SG1::call_din966
  DIN EN ISO 2010|dir1\bmp\DIN EN ISO 2010.bmp|F|MIP_SG1::call_iso2010
```

1) Comments and information
   This section is optional.
   As an example you can write here autor's name or any information you want to keep for yourself.
   If you write here anything – please make sure to start each line with the comment character '#'.

2) Menu Unique Name
   When SpaceLibrary reads a CCF File , it will create a menu with this name.
   Later , when we want to show this menu on screen, we simply call it by its name.
   Since you can process different CCF-Files for different menus, this name should be unique for each menu not to mix up menu names.

3) [About] section

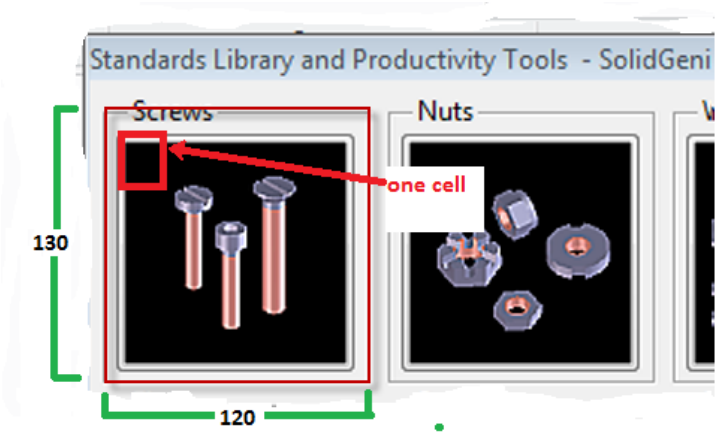


The line 'Powered by SpaceLibrary (2005 ..)' is automatically added and cannot be configured by the regular user.

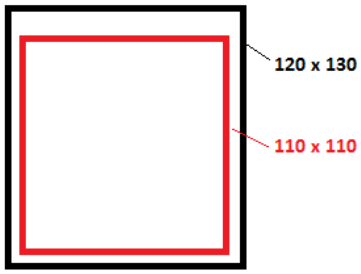
4) Size of a field (all fields in a menu have the same size)

Here we have 6 entries :

120 → menu field size in x in pixels
130 → menu field size in y in pixels
5 → one cell length
5 → one cell width
22 → for BMP picture, how many cells are in one field in X
22 → for BMP picture, how many cells are in one field in Y



In the above example, the reserved area for the BMP picture is 22 x 5 = 110 pixel in length and width.
The total area of the field is 120 x 130 pixels. That's why we have a small gap around.



5) Short Cut Definitions

Although short cuts are optional, they are a good way of keeping the CCF-File tidy, making the transfer of CCF-Files between computers easier and allowing the usage of Creo customizations folders in definitions.
Instead of writing full pathnames in the coming data section, you can once define a short cut and later use it in the definitions section.

E.g
All your pkg files are in the folder :
C:/Admin/MyData/parts/parts_for_ccf/new version/
Here you have files p1.pkg, p2.pkg, p3.pkg .. etc

Definition of files without short cut :
C:/Admin/MyData/parts/parts_for_ccf/new version/p1.pkg
C:/Admin/MyData/parts/parts_for_ccf/new version/p2.pkg
C:/Admin/MyData/parts/parts_for_ccf/new version/p3.pkg
…etc

Definition of files with a short cut :
SHORTCUT|dir1| C:/Admin/MyData/parts/parts_for_ccf/new version
dir1/p1.pkg
dir1/p2.pkg
dir1/p3.pkg
…etc

As seen above, the data entries are easier to read, the CCF File is tidier and at a later time if you want to change the folder of your pkg files – you only need to change the SHORTCUT definition.
Also if you give your CCF-File to your colleagues, they may have the pkg files in another folder.
Here they only have to modify the line with the short cut definition and all will work immediately.

Central location for parts and bmp files for all SG users and having same CCF File on each computer

You can also put all the pkg files in a central folder, map the folder location to a drive (Windows command 'Map network drive')
(here make sure that you assign the same drive letter for the network drive on each computer on the network)

E.g :
- all pkg files are on a network location //network_server/data/pkg files
- map network drive 'Y' to this folder
- define the drive 'Y' as SHORTCUT in the CCF file

SHORTCUT|dir1| Y: /electrical parts
SHORTCUT|dir2| Y: /mechanical parts

dir1/p1.pkg
dir1/p2.pkg
dir1/p3.pkg

dir2/mp1.pkg
dir2/mp2.pkg
dir2/mp3.pkg

Every person can use now the came CCF File without the need of local modifications.

You can define as many short cuts as you want.

There are 4 predefined keywords you can use in short cut definitions.

[installdir] → points to SolidGenius installation folder
sdcorpcustomizedir → points to the Creo customization folder 'corp'
sdsitecustomizedir → points to the Creo customization folder 'site'
sdusercustomizedir → points to the Creo customization folder 'user'

Examples :

SHORTCUT|dir1| sdcorpcustomizedir/pkg_files
SHORTCUT|dir2| sdcorpcustomizedir/bmp_files

6) Data Section

This is the most important section of CCF File.

Here we can define the menu structure and decide what will happen when the user hits a particular menu field ?

a) Field Definition

Even though each line can have a different function (create a sub menu, load pkg, load from ModelManager ..etc) they all have same structure:

Title of field | BMP-File | keyword | different parameters depending on the keyword

Keywords define what will happen when user hits this particular menu field.
There is a list of keywords you can choose from :

'S' … sub menu (when user hits this field, a sub menu will pop up)
'D' ... calls a dialog written in LISP
'F' ... calls a function written in LISP
'P' ... calls an external program
'LOS' ... loads a pkg file from file system
'LWM' ... loads a model from WorkManager (obsolent – not supported anymore)
'LMM' ... loads a model from ModelManager by its name only
'MMN' … loads a model from ModelManager by its name and from a specific CLASS
'MME' … loads a model from ModelManager by its ELID and from a specific CLASS
'T' ... to load a part from 'family of parts'

Every keyword needs different parameters.

Title of field | BMP-File | keyword | different parameters depending on the keyword

For example , to load a pkg file you should give the full pathname of the pkg file
To load a model from Modelmanager, you should give its name or ELID.

Here is a list of 'parameters for each keyword' and an example for its usage:

'S' … Title of sub menu when it pops up

```
Screws|dir1\bmp\Screws.bmp|S|Screws
```

'D' ... dialog name

```
Extrude|dir1\bmp\MyExtrude.bmp|D|MIP_SG1::my_extrude
```

'F' ... function name

DIN 963|dir1\bmp\DIN 963.bmp|F|MIP_SG1::call_din963

'P' ... program name or simply the filename if it is connected to a default programme

Notepad| C:\temp\file.bmp |P|notepad.exe

'LOS' ...full pkg file path

Creo Part|C:\MyParts\Test_ccf\part_1.bmp|LOS|C:\MyParts\Test_ccf\part_1.pkg
Creo Assembly|dir1\Test_ccf\assembly_1.bmp|LOS|dir1\Test_ccf\assembly_1.pkg

'LWM' ... WorkManager name (obsolent)

WM Part|dir1\Test_ccf\cylinder1.bmp|LWM|cylinder1

'LMM' ... model name in ModelManager

MM Part|dir1\Test_ccf\cylinder1.bmp|LMM|cylinder1

'MMN' … model name in ModelManager , CLASS name, Masterdata name

Name1| C:\temp\file.bmp |MMN|MM name|Class|Masterdata ELID
Name2| C:\temp\file.bmp |MMN|MM name|Class|

'MME' … ELID in ModelManager , CLASS name, Masterdata name

Name1| C:\temp\file.bmp |MME|MM Elid|Class|Masterdata ELİD
Name1| C:\temp\file.bmp |MME|MM Elid|Class|

'T' ...full path of table definition file

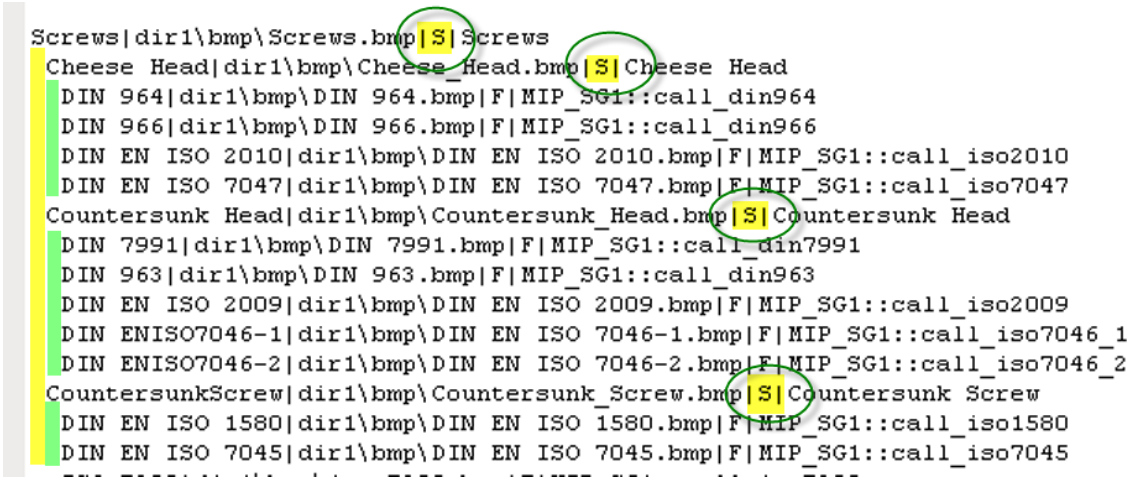
Table test|C:/temp/table.bmp|T|C:/temp/test_table.txt

b) Menu structure:

The menu structure is defined simply by the number of blank characters in the beginning of each data line.

The submenu definition line is built with 4 parameters separated by the '|' character in the following way :
The keyword is 'S' as 'sub menu'

Title of field | BMP-File | S | Title of menu when it opens up

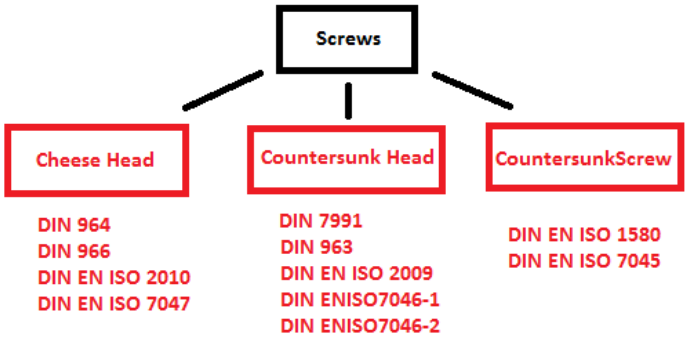


Look at the 2 different examples below :

Example 1

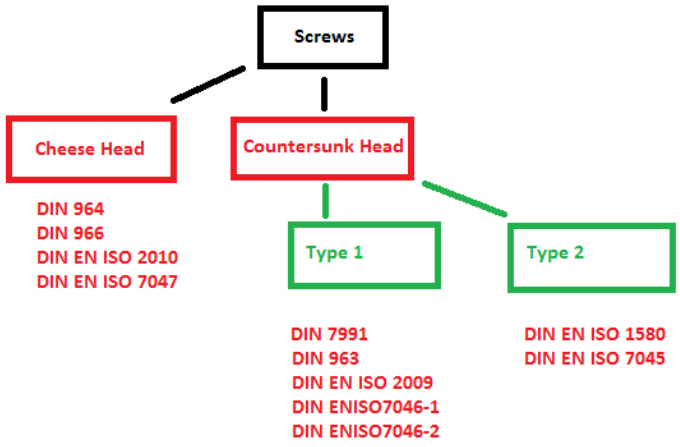
```
Screws|dir1\bmp\Screws.bmp|S|Screws
 Cheese Head|dir1\bmp\Cheese_Head.bmp|S|Cheese Head
  DIN 964|dir1\bmp\DIN 964.bmp|F|MIP_SG1::call_din964
  DIN 966|dir1\bmp\DIN 966.bmp|F|MIP_SG1::call_din966
  DIN EN ISO 2010|dir1\bmp\DIN EN ISO 2010.bmp|F|MIP_SG1::call_iso2010
  DIN EN ISO 7047|dir1\bmp\DIN EN ISO 7047.bmp|F|MIP_SG1::call_iso7047
 Countersunk Head|dir1\bmp\Countersunk_Head.bmp|S|Countersunk Head
  DIN 7991|dir1\bmp\DIN 7991.bmp|F|MIP_SG1::call_din7991
  DIN 963|dir1\bmp\DIN 963.bmp|F|MIP_SG1::call_din963
  DIN EN ISO 2009|dir1\bmp\DIN EN ISO 2009.bmp|F|MIP_SG1::call_iso2009
  DIN ENISO7046-1|dir1\bmp\DIN EN ISO 7046-1.bmp|F|MIP_SG1::call_iso7046_1
  DIN ENISO7046-2|dir1\bmp\DIN EN ISO 7046-2.bmp|F|MIP_SG1::call_iso7046_2
 CountersunkScrew|dir1\bmp\Countersunk_Screw.bmp|S|Countersunk Screw
  DIN EN ISO 1580|dir1\bmp\DIN EN ISO 1580.bmp|F|MIP_SG1::call_iso1580
  DIN EN ISO 7045|dir1\bmp\DIN EN ISO 7045.bmp|F|MIP_SG1::call_iso7045
```

The above data will create a menu structure as follows



Example 2

```
Screws|dir1\bmp\Screws.bmp|S|Screws
 Cheese Head|dir1\bmp\Cheese_Head.bmp|S|Cheese Head
  DIN 964|dir1\bmp\DIN 964.bmp|F|MIP_SG1::call_din964
  DIN 966|dir1\bmp\DIN 966.bmp|F|MIP_SG1::call_din966
  DIN EN ISO 2010|dir1\bmp\DIN EN ISO 2010.bmp|F|MIP_SG1::call_iso2010
  DIN EN ISO 7047|dir1\bmp\DIN EN ISO 7047.bmp|F|MIP_SG1::call_iso7047
 Countersunk Head|dir1\bmp\Countersunk_Head.bmp|S|Countersunk Head
  Type1|dir1\bmp\Countersunk_Head.bmp|S|Type1
   DIN 7991|dir1\bmp\DIN 7991.bmp|F|MIP_SG1::call_din7991
   DIN 963|dir1\bmp\DIN 963.bmp|F|MIP_SG1::call_din963
   DIN EN ISO 2009|dir1\bmp\DIN EN ISO 2009.bmp|F|MIP_SG1::call_iso2009
   DIN ENISO7046-1|dir1\bmp\DIN EN ISO 7046-1.bmp|F|MIP_SG1::call_iso7046_1
   DIN ENISO7046-2|dir1\bmp\DIN EN ISO 7046-2.bmp|F|MIP_SG1::call_iso7046_2
  Type2|dir1\bmp\Countersunk_Head.bmp|S|Type2
   DIN EN ISO 1580|dir1\bmp\DIN EN ISO 1580.bmp|F|MIP_SG1::call_iso1580
   DIN EN ISO 7045|dir1\bmp\DIN EN ISO 7045.bmp|F|MIP_SG1::call_iso7045
```



Note :

Please make sure that you are using backslash '\' instead of regular slash '/' in windows paths in CCF files.

(Regular slash '/' is a special character in LISP and will lead to errors if used in path definitions in CCF File)

Correct:

Creo Part|C:\MyParts\Test_ccf\part_1.bmp|LOS|C:\MyParts\Test_ccf\part_1.pkg

Wrong:

Creo Part|C:/MyParts/Test_ccf/part_1.bmp|LOS|C: /MyParts/Test_ccf/part_1.pkg

Can I call a PDF , EXCEL or WORD file from a menu field ?

If the file extension is defined in Windows to call a default program, then it is enough just to call the file by its name. Windows will automatically select the required program to open the file.
In this case – by using the keyword 'P' – you can easily call PDF, WORD or text editors like Notepad.

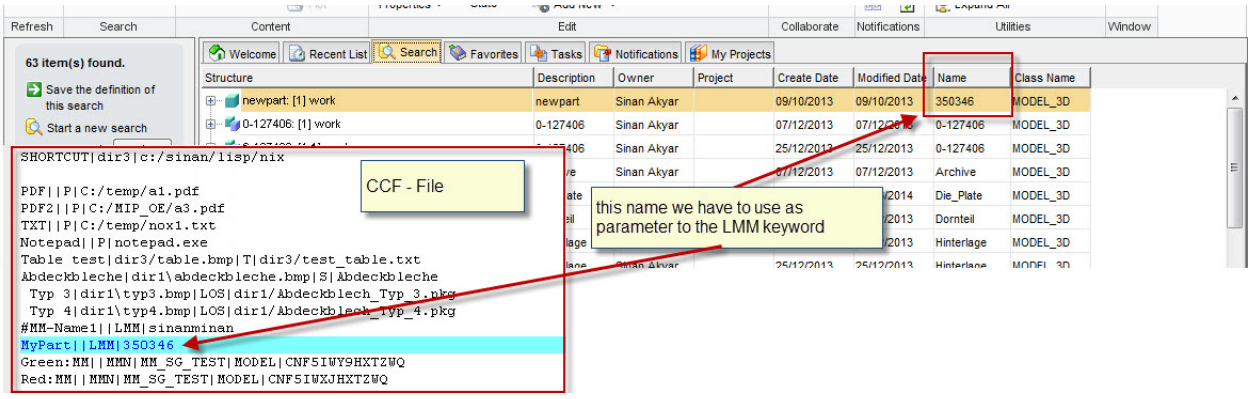
Examples :

PDF1|C:/temp/file.bmp |P|C:/temp/a1.pdf
PDF2| C:/temp/file.bmp |P|C:/MIP_OE/a3.pdf
TXT| C:/temp/file.bmp |P|C:/temp/nox1.txt
Notepad| C:/temp/file.bmp |P|notepad.exe

Additional information for 'Advanced Loading Keywords'

a) Keywords 'MMN' and 'MME' for ModelManager

LMM is the regular keyword to load a model from ModelManager.
The only parameter user can supply for LMM is the ModelManager name as seen below



But there are some limitations :

- if there are several versions of a model , the name will not be enough to determine which item should be loaded from ModelManager. Here user must use the ELID.

- LMM is searching models automatically in the class 'MODEL' . If the part is stored in another class than 'MODEL' , then keyword LMM will not be able to find it.

- If there are several MASTERDATA entries for a model , LMM command will automatically load the default MASTERDATA entry together with the model. It is not possible to select a MASTERDATA entry.

As seen in the above limitations, we need possibilities to define ELİD, CLASS and MASTERDATA options for more advanced loading possibilities

This can be done with CCF keywords MMN and MME :

<u>To load by name use MMN :</u>

Name1|bmpname|MMN|MM name|Class|Masterdata ELID

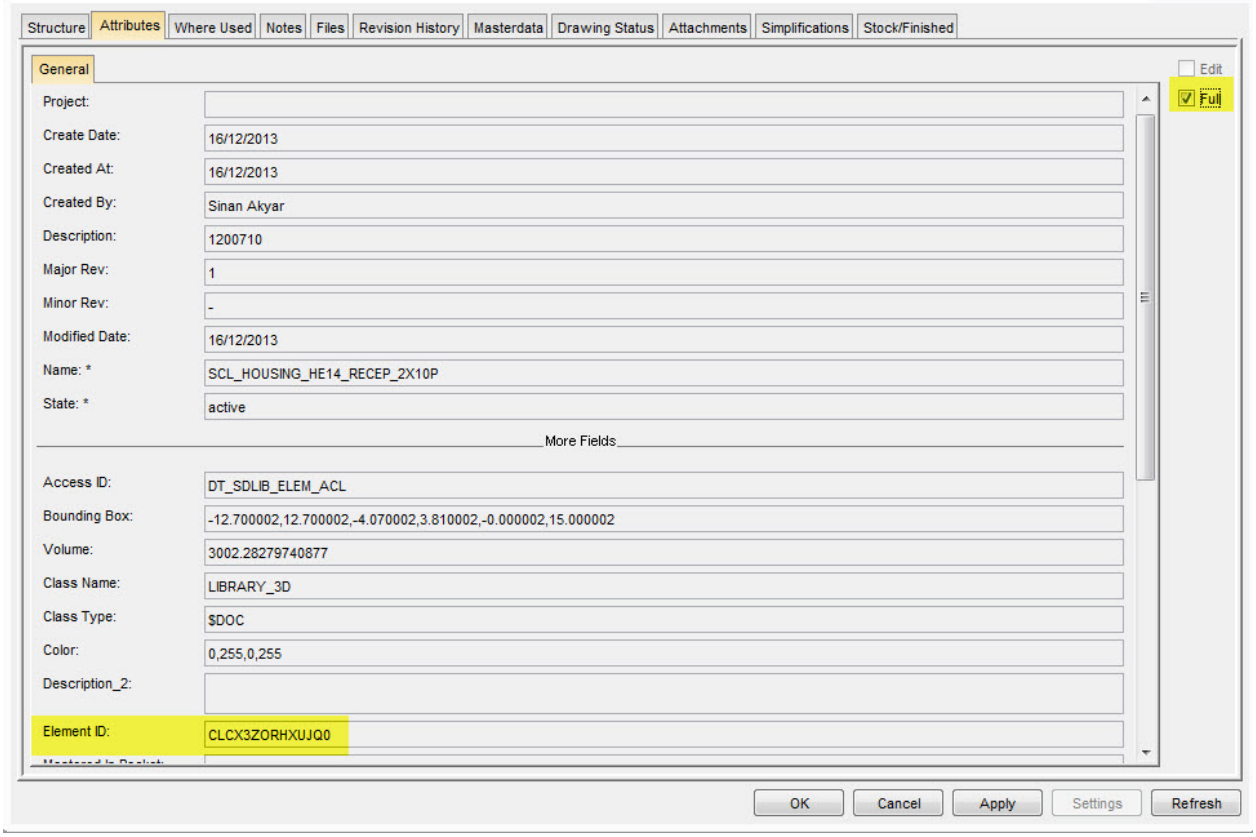
Name1|bmpname|MMN|MM name|Class|

<u>To load by ELID use MME :</u>

Name1|bmpname|MME|MM Elid|Class|Masterdata ELİD

Name1|bmpname|MME|MM Elid|Class|

The ELIDs of the model or of the Masterdata entries can found in the MM Attributes section by selecting the option 'Full'



Below you see some samples :

Loading from MODEL class using MM Name and a certain Masterdata entry :

Green_masterdata| my1.bmp |MMN|MM_SG_TEST|MODEL|CNF5IWY9HXTZWQ

Loading from MODEL class using MM Name

My Part| my1.bmp |MMN|MM_SG_TEST|MODEL|

Loading from MODEL class using MM ELID

My Part| my1.bmp |MME| CL9PHC3CZ8KFLT|MODEL|

Loading from MIP_3D_CLASS class using MM ELID

My Part| my1.bmp |MME| CL9PHC3CZ8KFLT| MIP_3D_CLASS |

Loading from MIP_3D_CLASS class using MM ELID and a certain Masterdata entry :

My Part| my1.bmp |MME| CL9PHC3CZ8KFLT| MIP_3D_CLASS | CNF7L0K1HXTZWQ

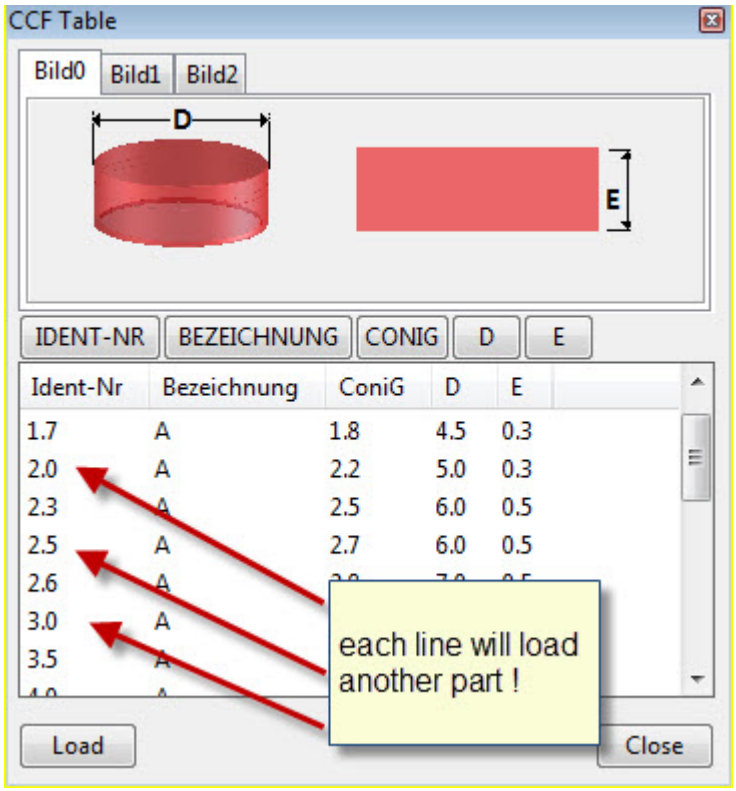
Loading from Library class using MM ELID

My Part| my1.bmp |MME| CL9PHC3CZ8KFLT library |

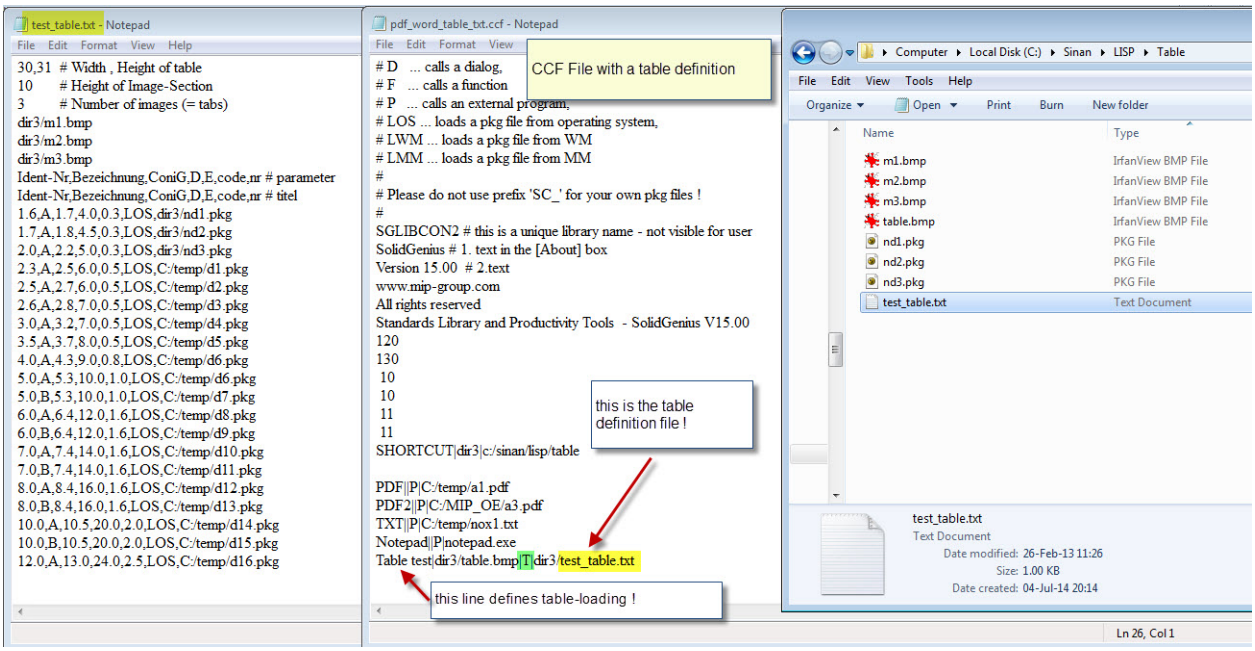
b) Keyword 'T' for family of parts

At certain times, you may have several parts you wish to handle with CCF- Loading.
If you would create a menu field for each part the menu would be too large and confusing.
In this case user can define menu field by using keyword 'T' which would pop up a table .

Below you see the definition of a table for 20 parts:

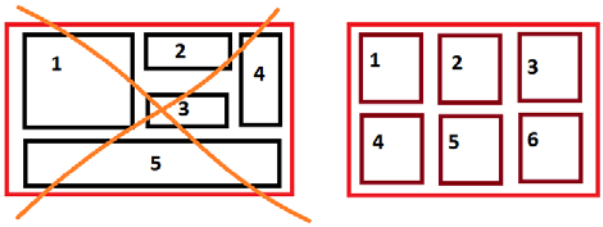


Below you see how a table is defined :

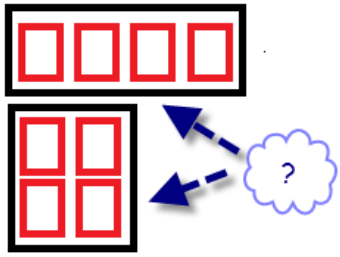


What are current limitations of CCF menus ?

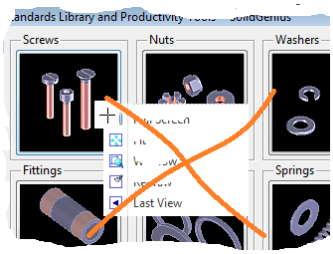
1) There is only one size entry in the CCF – file.
   This size is valid for the complete menu defined in this file.
   That's why it is not possible to have menu fields with different sizes inside one menu



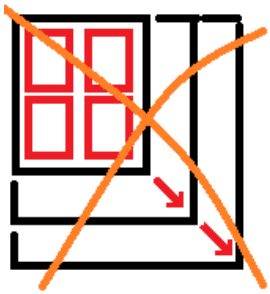
2) There is no parameter to define a 'menu form' inside CCF – File.
   Menu-Form is automatically decided at the generation time by SpaceLibrary



3) It is not possible to define a right-mouse-click menu inside individual menu fields



4) Menus are generated with a fix size. A resizing with the mouse is not possible.



DSG-FW-701-EN-01 08/2014