plug im ! Developer Guide

20191014

Plugins GUI configuration file : INI.xml

A plugin is a stand-alone executable with an xml file describing the parameter input interface, input and output file formats, as well as display or behavior options for plug im!. This file is in xml format, version 1.0 with "utf-8" encoding, and is named ExecutableNameINI.xml. It is composed as follows :

- Header xml

```
<?xml version="1.0" encoding="utf-8"?>
<Root>
```

- Plugin name, author and affiliation, quick description.

```
<Fonction Name="Analysis" Author="IFPEN 2013 MM" Desc="Analysis of connected components." />
```

- Descriptions of input and output files. The IN parameter (cf. ***Input file formats for plugins***) is used to specify the input file format. The OUTPREV and OUT parameters (cf. ***Output file formats for plugins***) are used to specify preview file(s) format and output file format respectivly.

```
<IN Type="bmp8" />
<OUT Type="BMP" />
<OUTPREV Type="BMP" />
```

- Directory name (*optional*). The plugin is accessible in a subsection of the plugin selection. Directory management is automatic: the first occurence of a directory name in an INI.xml file creates the subsection; occurences in another INI.xml files result in the placement in this subsection.

```
<Folder Name="Analysis" />
```

- Name of an export file (*optional*). An export file can be a file not managed natively by plug im ! or a second output file (cf. ***Output file formats for plugins***).

```
<Export Value="Analysis.txt" />
```

- Selection of the preview display mode (*optional*). There are 6 display modes : mode 1 (1) – Display of input data only , mode 2 (2) – Display of preview data only, mode 3 (1-2) – Side by side display, mode 4 (1|2) – Display cut input data first and then preview, mode 5 (2|1) – Display cut preview first and input data, mode 6 (1x2) – Mixed display with transparency. Default mode is mode 2.

```
<ModeShow Value="3" />
```

- Selection of the color LUT when first displaying the preview (*optional*). There are 7 color maps : 0 (STA) – 1 (NEG) – 2 (3D) – 3 (3Di) – 4 (MAT) – 6 (MATi) – 7 (Cycl). Default mode is the current color LUT.

```
<LUT Value="3" />
```

- Guidelines for specific display or behavioral actions of plug im ! (*optional*) (cf. ***Special action guidelines***).

```
  <SpecialAction AlwaysFitonPreview="true" AutoLaunchPreview="false"
HideShowMode="true"/>
```

- List of parameters necessary for the plugin to work (*optional*). The different parameters available under plug im ! are described below (cf. ***Plugins parameters***).

```
  <Param Type="Bool" Id="BoxCC" Name="Bounding box" Value="true"
Mode="Show or hide connected component bounding box." />

  <Param Type="Hide" Id="Langue" Name="English" Desc="Language for
results file (french or english)." />
```

- End of the xml file.

```
</Root>
```

Input file formats for plugins

plug im ! natively manages several file formats and can save a file as an input file of a plugin in the following format :

- **Signal**.
  - Cur, plug im ! internal format : `<IN Type="cur" />`

- **Image**.
  - Bitmap 8bits indexed : `<IN Type="bmp8" />`
  - Bitmap RGB 24bits : `<IN Type="bmpRGB" />`
  - Bitmap 8bits ou RGB 24bits : `<IN Type="bmp" />`
  - Portable Network Graphics, identical copy of the current screen visualization of plug im! : `<IN Type="png" />`
  - Tagged Image File Format 8 bits indexed : `<IN Type="tif8" />`
  - Tagged Image File Format RGB 24 bits : `<IN Type="tifRGB" />`
  - Tagged Image File Format integer 16 bits : `<IN Type="tif16" />`
  - Tagged Image File Format float 32 bits : `<IN Type="tif32" />`
  - Tagged Image File Format tous formats : `<IN Type="tif" />`
  - Tagged Image File Format tous formats image : `<IN Type="tif2D" />`
  - FDA (plug im! internal format) unsigned char 8 bits : `<IN Type="fda8" />`
  - FDA RGB 24bits : `<IN Type="fdaRGB" />`
  - FDA integer 32bits : `<IN Type="fdaI" />`
  - FDA float 32bits : `<IN Type="fdaF" />`
  - FDA double 64bits : `<IN Type="fdaD" />`
  - FDA complex 128bits : `<IN Type="fdaCX" />`
  - FDA tous formats : `<IN Type="fda" />`
  - FDA tous formats image : `<IN Type="fda2D" />`

- **Volume**.
  - Tagged Image File Format multi-pages 8 bits indexed : `<IN Type="tif3D8" />`
  - Tagged Image File Format multi-pages RGB 24 bits : `<IN Type="tif3DRGB" />`
  - Tagged Image File Format multi-pages integer 16 bits : `<IN Type="tif3D16" />`
  - Tagged Image File Format multi-pages float 32 bits : `<IN Type="tif3D32" />`
  - Tagged Image File Format tous formats : `<IN Type="tif" />`
  - Tagged Image File Format tous formats volume : `<IN Type="tif3D" />`
  - FDA (plug im! internal format) 3D unsigned char 8 bits : `<IN Type="fda3D8" />`
  - FDA 3D RGB 24bits : `<IN Type="fda3DRGB" />`
  - FDA 3D integer 32bits : `<IN Type="fda3DI" />`
  - FDA 3D float 32bits : `<IN Type="fda3DF" />`
  - FDA 3D double 64bits : `<IN Type="fda3DD" />`
  - FDA 3D complex 128bits : `<IN Type="fda3DCX" />`

- FDA 3D tous formats : `<IN Type="fda" />`
- FDA 3D tous formats volume : `<IN Type="fda3D" />`

- **Other**.
  - If the plugin does not need an input file : `<IN Type="" />`
  - If the module can take into account all files natively managed by plug im ! : `<IN Type="all" />`
  - plug im! can load without displaying any unsupported data types and make them available to the plugins. It is necessary to indicate the file extension in the IN tag parameter. For example, for an extension data « .ext » : `<IN Type=".ext" />`

Output file formats for plugins

plug im ! natively manages several file formats and can read the output file of a plugin in the following format :

- **Signal**.
  - Cur, plug im ! internal format :
    `<OUT Type="CUR" />`

- **Image**.
  - Bitmap format 8bits indexed, RGB (24bits, 32bits), ARGB/PARGB(32bits) (without recovery of the alpha layer) :
    `<OUTPREV Type="BMP" /> <OUT Type="BMP" />`
  - Portable Network Graphics format 8bits indexed, RGB (24bits, 32bits), ARGB/PARGB(32bits) (without recovery of the alpha layer) :
    `<OUTPREV Type="PNG" /> <OUT Type="PNG" />`
  - Tagged Image File Format 1bit or 8 bits indexed, Gray (8bits), RGB/BGR (24bits), RGBA/BGRA/PBGRA (32bits) (without recovery of the alpha layer, Gray16 (16bits), Gray32 (32bits) :
    `<OUTPREV Type="TIF" /> <OUT Type="TIF" />`
  - FDA (plug im! internal format) unsigned char 8 bits, RGB 24bits, integer 32bits, float 32bits, double 64bits, complex 128bits :
    `<OUTPREV Type="FDA" /> <OUT Type="FDA" />`

- **Volume**.
  - Tagged Image File Format multi-pages 1bit or 8 bits indexed, Gray (8bits), RGB/BGR (24bits), RGBA/BGRA/PBGRA (32bits) (without recovery of the alpha layer, Gray16 (16bits), Gray32 (32bits) :
    `<OUTPREV Type="TIF" /> <OUT Type="TIF" />`
  - FDA 3D (plug im! internal format) unsigned char 8 bits, RGB 24bits, integer 32bits, float 32bits, double 64bits, complex 128bits :
    `<OUTPREV Type="FDA" /> <OUT Type="FDA" />`

Plugins parameters

The plugin parameters are identified by a unique "Id" identifier. It is with this identifier that plug im ! will generate a PAR.xml parameter file used later by the plugin executable. It is possible to indicate for each parameter a brief description, and also whether it is accessible in normal or expert mode. The expert mode is reserved for parameters whose modifications can be considered as rare and/or complex. Each type of parameter has its own configuration options. There are also parameters that can be adjusted by user interaction, such as drawing areas of an image with the mouse, or selecting an abscissa by a vertical bar on a graph.

Basic options are the following :

- Name of the parameter displayed in the user interface: `Name="Number of peaks"`
- Accessibility in Expert mode only : `Expert="true"`
- Brief description : `Desc="Number of peaks to estimate."`

plug im ! natively handles these types of parameters :

- **Integer or Double**. Parameter of integer or real type defined with the keyword « Int » or « Double » respectivly, and a unique identifier.

```
<Param Type="Int" Id="intvar"
<Param Type="Double" Id="doublevar"
```

The available options are as follows :

- Name of the parameter : `Name="Variable A"`
- Default value : `Value="0"`
- Minimal of maximal values : `Min="-100" Max="10000"`.
- Unit increment value : `Incr="5"`
- Number of decimals (only for Double type) : `NbDec="2"`
- Increment speed when clicking and holding on the variation buttons + or -. If the option value is at `true`, the speed is constant, otherwise scrolling values accelerates over time which is the default behavior : `IncrLineaire="true"`
- Variation of values when right-clicking and moving along the vertical axis : `IsMouseUpDown="true"`
- Variation of values when right-clicking and moving along the horizontal axis : `IsMouseRightLeft="true"`
- Value cycling (exceeding the minimum or maximum value returns to the maximum or minimum respectively) : `Periodic="true"`

- **Double for signal data**. Real type parameter defined with the keyword « Double » and a unique identifier. When the input data is a graph, additional options are available.

```
<Param Type="Double" Id="doublevarsignal"
```

The additional options available are as follows:

- Automatic minimum and maximum values : `Min="auto" Max="auto"`. The parameter limits are then calculated according to the minimum and maximum of the input data.
- Default value based on automatic limits: `Value="A0.1"`. The default value is calculated by a fraction of the interval between the min and max limits.
- Selection of the value by a vertical bar on the graph : `XPosUserInteraction="true"` . The vertical bar is moved with a left click held with the mouse. The RGB color of the bar can be defined by the following three options: `colR="0" colB="200" colG="50"`. For an Expert parameter, the switch to Expert mode allows you to adjust the position of the bar by classic selection. To prevent the user from adjusting this position in Expert mode, indicate `XPosUILockIfExpert="true".`
- Selection of a hatched interval of mid-height interval. It is necessary that a first parameter with ID1 identifier of type Double allowing the selection by a vertical bar is present. The selection of a hatched interval is done with a parameter having as unique identifier: « ID1ITV » and the one per mid-height interval with a parameter with a unique identifier: « ID1MID ». The interval is set using the mouse wheel. For the mid-height interval, the mouse cursor must be close to the mid-height of the vertical bar.
  Here is an example where the parameter XMaxPic1 is used to set the position of a peak on a graph, the parameter XMaxPic1ITV is used to define an interval around this peak, and the parameter XMaxPic1MID is used to define a width at mid-height around the same peak:

```
    <Param Type="Double" Id="XMaxPic1" Name="Peak position 1"
Lineaire="false"                  Expert="true"         Value="A0.2"
XPosUserInteraction="true" colR="0" colB="200" colG="50" Incr="10"
Min="auto" Max="auto" NbDec="2" Desc="Maximum peak position."/>

  <Param Type="Double" Id="XMaxPic1ITV" Name="Deviation position"
Lineaire="false"        Expert="true"    Value="A0.1"    Incr="A0.01"
Min="0.00000001" Max="auto" NbDec="2" Mode="Normal" Desc="Maximum
deviation position."/>

  <Param Type="Double" Id="XMaxPic1MID" Name="1/2 width at half
height" Lineaire="false"     Expert="true" Value="0.1" Incr="10"
Min="0.00000001" Max="auto" NbDec="2" Mode="Normal" Desc="1/2 width
at half height."/>
```

- **Number of parameters**. This parameter allows you to display only a given number of parameters in the interface. It is defined with the keyword « Int » and a unique identifier.

```
<Param Type="Int" Id="itemnumber"
```

The parameters to be hidden/displayed must all be defined in the INI.xml file. The available options are the same as for type Int, with the two additional options that must be indicated:

- Number of parameters to be displayed by value increments: `LimitNextGUIItem="3"`
- Parameter shift from which the parameters are to be displayed : `DecalLimitNextGUIItem="6"`

Here is an example showing the use of a parameter of this type for several vertical bars allowing the definition of the position, interval and width at mid-height of a maximum of 2 peaks. The beginning of the definition of these peaks is located 1 parameter further than this parameter, the definition of a peak being carried out using 3 parameters :

```
  <Param Type="Int" Id="nbPics" Name="Number of peaks"
DecalLimitNextGUIItem="1" LimitNextGUIItem="3" Value="1"
Mode="Normal" Min="1" Max="2" Desc="Number of peaks to estimate."/>

  <Param Type="Bool" Id="AnalyseSensib" Name="Sensitivity analysis"
Value="true" Expert="true" Desc="Parameters sensitivity analysis."/>

  <Param Type="Double" Id="XMaxPic1" Name="Peak position 1"
Lineaire="false" Expert="true" Value="0.2"
XPosUserInteraction="true" colR="0" colB="200" colG="50" Incr="10"
Min="auto" Max="auto" NbDec="2" Desc="Maximum peak position."/>

  <Param Type="Double" Id="XMaxPic1ITV" Name="Deviation position"
Lineaire="false" Expert="true" Value="0.1" Incr="10"
Min="0.00000001" Max="auto" NbDec="2" Mode="Normal" Desc="Maximum
deviation position."/>

  <Param Type="Double" Id="XMaxPic1MID" Name="1/2 width at half
height" Lineaire="false" Expert="true" Value="0.1" Incr="10"
Min="0.00000001" Max="auto" NbDec="2" Mode="Normal" Desc="1/2 width
at half height."/>

  <Param Type="Double" Id="XMaxPic2" Name="Peak position 2"
Lineaire="false" Expert="true" Value="0.3"
XPosUserInteraction="true" colR="0" colB="200" colG="50" Incr="10"
Min="auto" Max="auto" NbDec="2" Desc="Maximum peak position."/>

  <Param Type="Double" Id="XMaxPic2ITV" Name="Ecart position"
Lineaire="false" Expert="true" Value="0.1" Incr="10"
Min="0.00000001" Max="auto" NbDec="2" Mode="Normal" Desc="Maximum
deviation position."/>

  <Param Type="Double" Id="XMaxPic2MID" Name="1/2 width at half
height" Lineaire="false" Expert="true" Value="0.1" Incr="10"
Min="0.00000001" Max="auto" NbDec="2" Mode="Normal" Desc="1/2 width
at half height."/>
```

- **Bool**. Boolean type parameter defined with the keyword "Bool" and a unique identifier.

```
<Param Type="Bool" Id="boolvar"
```

  - Default value is « true » or « false » : `Value="true"`

This parameter can also be used to hide or display another parameter. It is then necessary to add in the options of the other parameter the following keywords :

  - To display the parameter according to a Boolean parameter from boolvar identifier to true: `VisibleIfBoolId="boolvar"`
  - To hide the parameter according to a Boolean parameter from boolvar identifier to true: `NoVisibleIfBoolId="boolvar"`

Here is an example showing the use of a Boolean parameter allowing if it is true to display an integer type parameter.

```
<Param Type="Bool" Id="AjustParam" Name="Adjust parameter" Value="false" Desc="If check, user can adjust special parameter."/>

<Param Type="Int" Id="SpeParam" Name="Special parameter" VisibleIfBoolId="AjustParam" Value="10" Min="10" Max="10" Desc="Special parameter."/>
```

- **Multiple choice list**. Parameter allowing to choose a character string from a list of character strings. It is defined with the keyword « LStr » and a unique identifier.

```
<Param Type="LStr" Id="lstrvar"
```

The additional options available are as follows:

  - List of strings, separated by the character '|' : `ListeStr="Choix 1|Choix 2|Troisième choix"`
  - The default value is one of the choices in the list defined in ListeStr : `Value="Choix 2"`
  - In case you want a display in the user interface different from the choice list retrieved by the plugin, it is possible to use `ListeStrGUI`. The list displayed then follows the same order as the list defined in `ListeStr`, it must also have the same number of choices : `ListeStrGUI="Case 1|Case 2|Case 3"`
  - By default, the list is sorted in alphabetical order. To keep the initial order: `ForceNoSortOrder="true"`
  - Automatically fill the list with the names of the modules available under plugim ! : `ListeStr="SPECIALPLUGINLIST"`. `ListeStrGUI` is then useless.

A choice list can also be used to hide or show another parameter. It is then necessary to add in the options of the other parameter the following keywords:

  - To display the parameter if a given choice is selected in the list : `VisibleIfLStrId="lstrvar" VisibleIfLStrValue="Particular case"`
  - To hide the parameter if a given choice is selected in the list : `NoVisibleIfLStrId="lstrvar" VisibleIfLStrValue="Particular case"`

Here is an example showing the use of a choice list parameter allowing if the choice « Case A » is chosen to display an integer parameter type.

```
<Param Type="LStr" Id="ListParam" Name="Case" Value="Case A" ListeStr="Case A|Case B|Case C" Desc="Case selection."/>

<Param Type="Int" Id="SpeParam" Name="Special parameter" VisibleIfLStrId="ListParam" VisibleIfLStrValue="Case A" Value="10" Min="10" Max="10" Desc="Special parameter."/>
```

- **File selection**. Parameter allowing to select a file through a system window. It is defined with the keyword « FSelect » and a unique identifier.

```
<Param Type="FSelect" Id="fselectvar"
```

The additional options available are as follows:

- The default value is used to specify a default file. Without a full path, by default the file is considered as present in the plugin directory: `Value="file.bmp"`
- Filter file formats, uses standard system syntax: `Filter="Bitmap file|*.bmp|PNG file|*.png|All Files|*.*"`

- **Save file selection**. Parameter allowing to select or create a file to be saved through a system window. It is defined with the keyword « SSelect » and a unique identifier.

```
<Param Type="SSelect" Id="saveselectvar"
```

The additional options available are as follows:

- The default value allows you to specify a default file name : `Value="file.bmp"`
- Filter file formats, uses standard system syntax: `Filter="Bitmap file|*.bmp|PNG file|*.png|All Files|*.*"`

- **Directory selection**. Parameter allowing to select a folder through a system window. It is defined with the keyword « DSelect » and a unique identifier.

```
<Param Type="DSelect" Id="dirselectvar"
```

The additional options available are as follows:

- The default value allows you to specify a default folder, an empty value places the default folder in the plugin folder: `Value="C:\Temp\"`
- To use the standard Windows directory selection dialog box: `OldStyleDirectorySelection="true"`

- **Hidden parameter**. Hidden parameter not accessible by the user interface. It is defined with the keyword « Hide » and a unique identifier.

```
<Param Type="Hide" Id="hiddenvar"
```

- Default value : `Value="defaultvar"`

plug im ! also manages input parameters with user interaction on the data. It is possible to use only one of these parameters per plugin.

- **Draw on image**. Parameter allowing to draw with the mouse on an image. The drawing is done by holding down the right button. A menu on the left in the plug im! interface is available to hide/display the drawing and define the size and color of the brush. It is also possible to delete (color E). There is no unique identifier and a file is created by plug im! whose path and name can be retrieved using a specific tag in the file *PAR.xml (see additional options).

```
<Param Type="OverlayDraw"
```

  - Type of image format representing the drawing in bmp, png or fda format recoverable by the tag MaskBMP, MaskPNG or MaskFDA respectivly:
    `Ext="bmp"`
  - Number of brush colors from 1 to 3 (default 3):
    `nbMark="2"`
    Value equal to 1 : one color (color M r : 255, g : 69, b : 0)
    Value equal to 2 : one color and an eraser (color E r : 0 g : 0 b : 0)
    Value equal to 3 : one color, an eraser, and a second colors (color M' r : 75, g : 0, b : 130)
  - Size of the brush (default 10):
    `TMark="15"`

- **Line on image**. Parameter allowing to draw a segment with the mouse on an image. The drawing is done by holding down the right button. There is no unique identifier to indicate and a file is created by plug im! whose path and name can be retrieved using the tag « MaskTXT » in the file *PAR.xml (see additional options). This file lists on one line the x and y coordinates in pixels of the ends E_1 and E_2 of the segment : « xE_1 yE_1 xE_2 yE_2 ».

```
<Param Type="OverlayLine"/>
```

- **Vertical line on image**. Parameter allowing to draw a vertical line with the mouse on an image. The drawing is done by holding down the right button or by clicking. There is no unique identifier to indicate and a file is created by plug im! whose path and name can be retrieved using the tag « MaskTXT » in the file *PAR.xml (see additional options). This file lists on one line the width position in pixels of the line: « x ».

```
<Param Type="OverlayLineVer"/>
```

- **Horizontal line on image**. Parameter allowing to draw a horizontal line with the mouse on an image. The drawing is done by holding down the right button or by clicking. There is no unique identifier to indicate and a file is created by plug im! whose path and name can be retrieved using the tag « MaskTXT » in the file *PAR.xml (see additional options). This file lists on one line the height position in pixels of the line: « y ».

```
<Param Type="OverlayLineHor"/>
```

- **Point on image**. Parameter allowing to place o point with the mouse on an image. The position is set by click down the right button. There is no unique identifier to indicate and a file is created by plug im! whose path and name can be retrieved using the tag « MaskTXT » in the file *PAR.xml (see additional options). This file lists on one line the x and y coordinates in pixels of the point : « x y».

```
<Param Type="OverlayPoint"/>
```

- **Circle on image**. Parameter allowing to draw a circle with the mouse on an image. The drawing is done by holding down the right button, starting from the center of the circle. There is no unique identifier to indicate and a file is created by plug im! whose path and name can be retrieved using the tag « MaskTXT » in the file *PAR.xml (see additional options). This file lists in pixels on one line the x and y coordinates of the center C of the circle, then its radius r : « xC yC r »

```
<Param Type="OverlayCircle"/>
```

- **Box on image**. Parameter allowing to draw a rectangle with the mouse on an image. The drawing is done by holding down the right button, starting from one of the corners of the rectangle. There is no unique identifier to indicate and a file is created by plug im! whose path and name can be retrieved using the tag « MaskTXT » in the file *PAR.xml (see additional options). This file lists in pixels on one line the x and y coordinates of the upper left corner of the rectangle then the width L and height H: « x y L H »

```
<Param Type="OverlayBox"/>
```

Special action guidelines

It is possible to specify certain display or behavior options to plug im! These features are to be specified in the INI.xml file of the plugin, using the keyword « SpecialAction » (cf. ***Plugins GUI configuration file : INI.xml***) :

```
  <SpecialAction AlwaysFitonPreview="true" AutoLaunchPreview="false"
HideShowMode="true"/>
```

If not present, these features are used with their default value.

The list of these features is as follows:

- **Graphical behaviour**
  - Display the plugin's graphical window (by default to « false ») : `CreateNoWindows="true"`
  - Handle graphics as images, i.e. do not recalculate the graph for each resizing in display mode « cut » 4 (1|2) ou 5 (2|1) (by default to « false ») : `SplitModeGraphicAsImage="true"`
  - Scale the data to the screen when the first preview is calculated (by default to « false ») : `FitonPreview="true"`
  - Set the data to its initial size when the first preview is calculated (by default to « false ») : `OneonPreview="true"`
  - Systematically scale the data to the screen each time the preview is calculated (by default to « false ») : `AlwaysFitonPreview="true"`
  - Systematically set the data to its initial size each time the preview is calculated (by default to « false ») : `AlwaysOneonPreview="true"`
  - Scale the data to the screen if you leave the module without executing it (« Back » button) (by default to « false ») : `FitonBack="true"`
  - Set the data to its initial size if you leave the module without executing it (« Back » button) (by default to « false ») : `OneonBack="true"`
  - Do not display the preview directly (by default to « true ») : `AutoLaunchPreview="false"`

- **Recording user actions**
  - Save each navigation change in the 3D data (by default to « false ») : `Update3Dchange="true"`
  - Save each color palette change (by default to « false ») : `UpdateLUTchange="true"`
  - Save each change of scale change (by default to « false ») : `UpdateScalechange="true"`
  - Save each modification of image or graphic move (by default to « false ») : `UpdateOffsetchange="true"`

- **Block / activate automatic actions**
  - Block the automatic execution of the plugin each time a parameter is changed (by default to « true ») : `RestartIfUpdate="false"`
  - Restart the plugin execution when clicking on the execute button (by default to « false ») : `RestartIfExecute="true"`

- Restart the plugin execution when switching to preview mode (by default to « false ») : `RestartIfPreview="true"`
- Do not return to the module selection list after the module has been executed and stay on the current module (by default to « true ») : `ExitIfExecute="false"`
- Reset the module to its default settings when selected (by default to « false ») : `AlwaysDefaultParamOnStart="true"`

- **Block / activate functionnalities**
  - Block image or graphic resizing with the mouse wheel (by default to « false ») : `StopMouseScale="true"`
  - Block images or graphics moves with the mouse (by default to « false ») : `StopMouseMove="true"`
  - Hide the list of choices for the preview display mode (by default to « false ») : `HideShowMode="true"`
  - Hide the checkbox to start the preview calculation (by default to « false ») : `HidePreview="true"`

- **Features related to the linguistic format of the execution machine**
  - Force the saving of parameter files *PAR.xml with a point as decimal separator (useful for plugin compiled in Fortran) (by default to « false ») : `ForceDecimalDotPAR="true"`
  - Force the saving of *.cur graphic files with a dot as decimal separator (useful for modules compiled in Fortran) (by default to « false ») : `ForceDecimalDotCUR="true"`

- **Additional features and functions**
  - Force plug im ! to consider the final result of a module (save with the tag « OUT ») as a preview result (useful if creating a preview file different from the final result is not necessary) (by default to « false »): `FinalResultIsPreview="true"`
  - Enable clipboard management for fast preview file recovery (cf. ***Plugin console message guidelines*****)** (by default to « false ») : `CheckClipboardImage="true"`

Plugin console message guidelines

It is possible to ask plug im! to perform certain actions from a message in the console. By default, a message in the console written by a module is taken over and displayed in the plug im ! user interface. It is possible by using certain keywords to launch specific actions :

- **Data display.**
  - Scaling data to the screen:
    `SPECIAL SCALETOFIT`
  - Scaling data to original size:
    `SPECIAL SCALETOONE`
  - Shift the data horizontally by a certain number of pixels X:
    `SPECIAL Xoffset X`
  - Shift the data horizontally by a certain number of pixels Y. The data orientation is the "image" orientation, the direction of the Y axis being downwards:
    `SPECIAL Yoffset Y`

- **Show a message.**
  - Display of a message in the middle of the interface:
    `SPECIAL MESSAGE text…`

- **Load a new data.**
  - Load a data instead of the existing data:
    `SPECIAL LOAD0 filename`

- **Modifying the value of a parameter.**
  - Update the value of the Idparam identifier parameter of the module with the value val:
    `SPECIAL PARAM Idparam val`

Description de l’aide, avec les paramètres spéciaux de l’interface

Tag spéciaux dans les PAR.xml

Description des fichier CUR

Description des fichier FDA