MDL Chime™ Embed Tag Options Reference for 2.6 SP5

The HTML developer inserts a MDL Chime plug-in into an HTML page via an EMBED tag. The following options are used to control the embedded MDL Chime plug-in:

Alphabetical Listing of the Embed Tags

Listing by Category

altscript

Specifies alternate RasMol commands to execute when a button goes from a "pushed" state to an "unpushed" state. Multiple commands can be separated with "|" or ";". This tag applies only to toggle or radio buttons, as specified in the **button** tag.

Syntax

altscript={RasMol script commands}

See also

button
buttonstate
script

Example

```
<!-- the first Chime displays the structure -->
<embed src="3dfr.pdb" name="themol"
width=400 height=350
spinX=0 spinY=0 spinZ=50 spinfps=30
><br>
<!-- this button toggles between spinning and not -->
<embed type="application/x-spt" width=20 height=20
button="toggle" target="themol"
script="spin" altscript="spin off"
>
```

---

animfps

Specifies the animation events (frames) per second.

Note: The value specified for **animfps** may not be possible on the viewer's platform. Chime will get as close as the system allows.

Syntax

animfps={animation events per second}

See also

AnimFrameCallback
animmode
startanim

Example

```
<embed src="sn2.xyz" display3d=ball&stick width=300 height=200
animfps=30 startanim=true animmode=loop
>
```

AnimFrameCallback

Specifies the name of a JavaScript function in the current document that is to be called before each event ("frame") of an animation (.xyz) file is displayed. The function should conform to the prototype:

```
function JSFunctionName( pluginName, frameNumber )
```

Where *pluginName* is the value specified in the **name** parameter for the Chime structure window in which the animation is playing and *frameNumber* is an integer indicating which number event in the file is about to be displayed.

Syntax

AnimFrameCallback={JSFunctionName}

See also

animfps
animmode
startanim

Example

```
<script language="JavaScript">
//This callback function reports which frame is being displayed
function JSAnimcontrol( PlugName, FrameNo ) {
  document.form1.messageboard.value = "Now showing frame " + FrameNo;
}
</script>
<embed src="sn2.xyz" name="snchime" width=350 height=325
    display3d=ball&stick color3d=cpk options3d=specular
    animfps=30 startanim=true animmode="loop" AnimFrameCallback="JSAnimcontrol"
>
<form name=form1>
<input name="messageboard" size=20>
</form>
```

---

animmode

Specifies how the the animation is to be presented.

Syntax

animmode={loop|once|palindrome|ping|pong}

loop - Play the animation from the first frame to the last and then start again from the first.

once - Play the animation from the first frame to the last and then stop.

palindrome - Play the animation from the first frame to the last and then backwards from last to first, continuously.

ping - Synonym for **palindrome**.

pong - Synonym for **palindrome**.

See also

animfps
AnimFrameCallback
startanim

Example

```
<embed src="sn2.xyz" display3d=ball&stick width=300 height=200
animfps=30 startanim=true animmode=loop
>
```

---

atommapnum2d

An atom-atom map on the reaction components specifies exactly which atoms in the reactants correspond to the atoms in the products. Applies to reactions in 2D.

Syntax

atommapnum2d={off|on}

off – do not display atom-atom maps

on - display atom-atom maps

See also

invretmarks2d
reactingcenters2d

---

atomnum2d

Specifies whether to display atom numbers.

Syntax

atomnum2d={on|off}

on - display atom numbers

off - do not display atom numbers.

See also

bondscale2d
bondlen2d
fontsize2d
fontname2d
hlabels2d

---

bgcolor

Sets the background color for either 2d or 3d rendering.

Syntax

bgcolor={black|white|#rrggbb}

black - make the background black

white - make the background white

#rrggbb - set the background to a specific color using an HTML-style color value. For example: #777777 gives a dark gray.

See also

color2d
color3d
palette

---

bondlen2d

Specifies the standard bond length, in decipoints, used to display a 2D image when using the tag **bondscaling=stdbond**. The default length is 180.

Syntax

bondlen2d={###}

See also

bondscale2d

---

bondscale2d

Specifies the type of scaling to use when displaying a 2D image.

Syntax

bondscale2d={asdrawn|asdrawn_fitbox|fitbox|stdbond}

asdrawn - display the image exactly as it was drawn, with the same bond lengths and coordinates.

asdrawn_fitbox - display the image with the same relative bond lengths as were drawn, but scale the whole image to best fit the Chime structure window.

fitbox - display the image to best fit the Chime structure window, and make all bonds equal in length.

stdbond - set all bond lengths to the standard bond length. The standard bond length is specified by the **bondlen** tag.

See also

atomnum2d
bondlen2d
fontsize2d
fontname2d
hlabels2d

---

button

Displays the Chime plug-in as a button. Three types of buttons are available: push, radio and toggle.

When the user clicks a button, the script attached to the button through the **`script`** or **`csml`** tag executes. The script acts on the Chime plug-in indicated by the **`target`** tag. A Chime button can also trigger a JavaScript routine through the **`ButtonCallback`** tag.

In the case of toggle and radio buttons, a second script can be activated when the button goes from a "pushed" to an "unpushed" state. This script is specified by the **`altscript`** tag.

Because a Chime button does not display a file, you must specify the mime type manually using the `type` tag: `type="application/x-spt"`, as shown in the example.

Syntax

`button={push|radio#|toggle}`

`push` - a simple push-button.

`radio#` - a member of a radio group. Only one member of the group is pushed at any given time. The group is identified by the number attached to the tag, for example "radio1", "radio2".

`toggle` - a toggle button which alternates between a "pushed" state and an "unpushed" state, similar to a check box.

See also

altscript
ButtonCallback
buttonstate
csml
immediate
target

Example

```
<!-- the first Chime displays the structure -->
<embed type="chemical/x-mdl-molfile" width=400 height=350
src="surface/1hew.pdb"
name="themol"
spiny=80
startspin=true
><br>
<!-- the second is a button that affects the displayed structure -->
Toggle Transparency
<embed type="application/x-spt" width=15 height=15
script="list aminoacid transparent"
button=toggle
target=themol
>
```

---

buttonstate

Used to specify that a button's initial state is "pushed". If this tag is not used, the button's initial state will be "unpushed".

Syntax

buttonstate={pushed}

See also

altscript
button

ButtonCallback

This tag is only to be used in a Chime button (**button**=true). It specifies a JavaScript function in the current document that is to be called whenever the button is pushed. The function is called twice, once before execution of the attached RasMol or CSML script, and a second time afterwards. If there is no attached script, the function is still called twice.

The JavaScript function must conform to the prototype:

```
function JSFunctionName( pluginName, executedYet? )
```

...where *pluginName* is the name of the Chime button which the user has pressed, and the boolean *executedYet?* is false when JsFunctionName is first called (before execution of the script) and true the second time.

With Chime 2.0's new LiveConnect interface, you can start RasMol scripts from a JavaScript routine when running in Netcape Navigator, as shown in the example below. In other browsers, you can achieve the same result by writing a new button with the **immediate** tag.

Syntax

ButtonCallback={JSFunctionName}

See also

button
csml
immediate
script

Example

```
<!-- this routine toggles the spin using the LiveConnect interface -->
<script language="JavaScript">
function SpinMol(pluginName, executedYet) {
  if (!executedYet) { //make sure only executes once per click
    if (spinning) {
      document.themol.executeScript("spin false");
      spinning = false;
    } else {
      document.themol.executeScript("spin");
```

```
      spinning = true;
    }
  }
}
spinning = false;
</script>
<!-- the first Chime displays the structure -->
<embed type="chemical/x-mdl-molfile" width=400 height=350
src="surface/1hew.pdb"
name="themol"
spiny=80
startspin=false
><br>
<!-- the second is a button that executes the JavaScript routine -->
<embed type="application/x-spt" width=30 height=30
button=push
ButtonCallback=SpinMol
>
```

---

color2d

Specifies the foreground color for a 2D rendering.

Syntax

color2d={black|sketch|white|#rrggbb}

black - make the foreground black

sketch - when displaying a sketch file, use the colors specified in the sketch instead of overriding with a standard color

white - make the foreground white

#rrggbb - set the foreground to a specific color using an HTML-style color value. For example: #777777 gives a dark

gray.

See also

bgcolor
display2d

---

color3d

Specifies the color scheme for 3D display.

Syntax

color3d={chain|cpk|group|monochrome|shapely|structure|temperature|user}

The options correspond to those available in the Chime menu.

See also

bgcolor
display3d
options3d
palette
scale3d

---

csml

Specifies any valid CSML script commands to apply to the plug-in. Multiple commands can be separated with a "|" or ";"

When attached to a Chime structure window, the script executes immediately when the plug-in loads. When attached to a Chime button (**button**=true), the script executes in the Chime structure window indicated by the **target** tag when the button is pushed.

Syntax

csml={valid CSML script commands}

See also

button
immediate
script
target

---

debugscript

When **debugscript** = true, each line of an executing RasMol script is echoed to the browser's status line.

Syntax

debugscript={false|no|true|yes}

false or **no** - do not echo script commands

true or **yes** - echo script commands to status line

See also

script

display2d

Forces a 2D rendering of a molecule that would be rendered in 3D by default. Any molecule with 3D coordinates is automatically rendered in 3D unless this tag is used.

Syntax

```
display2d=true
```

See also

color2d
display3d

---

display3d

Specifies the type of 3D display.

Syntax

```
display3D={backbone|ball&stick|cartoons|ribbons|spacefill|sticks|strands|wireframe}
```

The options correspond to those available in the Chime menu.

See also

color3d
display2d
options3d
scale3d

---

expandResidues2d

Specifies whether to display expanded or contracted residues. The default display of residues in Chime Pro is the contracted form (in Chime Pro 2.0 the default was the expanded form).

Syntax

expandResidues2d = {on|off}

on - Display expanded residues

off - Do not display expanded residues (default)

---

fontname2d

Sets the font used to display atom symbols and other text in 2D display. The default in Windows is Arial.

Syntax

fontname2d={font name}

See also

atomnum2d
bondscale2d
bondlen2d
fontsize2d
hlabels2d

---

fontsize2d

Sets the font size, in decipoints, used to display atom symbols and other text in 2D display. The default size is 120.

Syntax

```
fontsize2d={###}
```

See also

atomnum2d
bondscale2d
bondlen2d
fontname2d
hlabels2d

---

frank

When **`frank`** `= true`, the "MDL" trademark is displayed in the lower right corner of the Chime structure window.

Syntax

frank={false|no|true|yes}

---

hbonds2d

Specifies whether, and how, to display hydrogen bonds.

Syntax

hbonds2d={off|on|number}

off - do not display hydrogen bonds

on - display hydrogen bonds as dashed lines

number - display hydrogen bonds as cylinders whose diameter is proportional to the number. See example below.

See also

ssbonds

Example

```
<embed src="1prc.pdb" height=600 width=600 hbonds=20>
```

---

hlabels2d

Specifies how to display implicit hydrogens in a 2D structure.

Syntax

hlabels2d={asdrawn|false|hetero|terminalhetero|true}

asdrawn - display hydrogens as originally drawn.

false - display no implicit hydrogens.

hetero - display implicit hydrogens only on hereoatoms.

terminalhetero - display implicit hydrogens only on terminal and hetero atoms.

true - display implicit hydrogens on all atoms.

See also

atomnum2d
bondscale2d
bondlen2d
fontsize2d
fontname2d

---

immediate

Specifies whether the script attached to a Chime button should be run automatically when the Chime is loaded. This feature allows JavaScript to communicate with Chime by writing a Chime button (to an unseen part of the screen) that will execute immediately to perform some operation on a Chime structure window.

NOTE: Chime 2.0 supports a LiveConnect interface in Netscape Navigator, providing an easier way to run RasMol scripts from JavaScript. See **button** for an example.

Syntax

immediate={false|no|true|yes}

false or **no** - do not run script immediately. Script will run only when button is pushed.

true or **yes** - run script immediately upon loading Chime.

See also

button
csml
preloadscript
script
target

Example

```
////////////////////////////////////////////////
//      JavaScript function sendRasmolScriptToChime
//      Developed by Eric Martz, U Mass (Amherst)
//      http://www.umass.edu/microbio/chime/chimehow/chime_3.htm
//      'Sends' scripts to chime by creating invisible chime button
//      with the 'immediate' embed tag, which causes the script
//      to be sent to chime at the creation of the button
////////////////////////////////////////////////
function sendRasmolScriptToChime( targetchimename, script )
{
// write a Chime button which will execute it's script immediately
        top.fordummychimes.document.open();
        top.fordummychimes.document.writeln("<html><head></head><body>");
        top.fordummychimes.document.writeln("<embed type=\"application/x-spt\" hidden=true ");
        top.fordummychimes.document.writeln("width=10 height=10 button=push target=\'"+targetchimename+"\' '
        top.fordummychimes.document.writeln("script=\'"+script+"\' immediate=1>");
        top.fordummychimes.document.writeln("</body></html>");
        top.fordummychimes.document.close();
}
```

invretmarks2d

Inversion marks specify that a stereo center must be inverted by the reaction. Retention marks specify that a stereo center must retain its configuration. Applies to reactions in 2D.

Syntax

invretmarks2d={off|on}

off – do not display the stereochemical changes that are apparent in reactions

on - display the stereochemical changes that are apparent in reactions

See also

atommapnum2d
reactingcenters2d

---

jcamp_blocknum

For BLOCK or NTUPLE JCAMP-DX files specifies which data set to initially display.

Syntax

jcamp_blocknum={1...n}

Integer value – display that data set

See also

jcamp_grid
jcamp_revplot
jcamp_help

---

jcamp_grid

Specifies whether to display a grid on the JCAMP-DX data.

Syntax

`jcamp_grid={false|true}`

`false` – do not display grid

`true` – display grid

See also

jcamp_blocknum
jcamp_revplot
jcamp_help

---

jcamp_help

Specifies whether to turn on/off the JCAMP-DX interpret mode.

Syntax

`jcamp_ help ={false|true}`

`false` – set interpret off

`true` – set interpret on

See also

jcamp_blocknum
jcamp_grid
jcamp_revplot

---

jcamp_revplot

Specifies whether to reverse the JCAMP-DX plot

Syntax

`jcamp_ revplot ={false|true}`

`false` – do not reverse JCAMP-DX plot

`true` – reverse JCAMP-DX plot

See also

jcamp_grid
jcamp_help

LoadStructCallback

Specifies the name of a JavaScript function in the current document that is to be called whenever the user pastes or transfers a structure or opens a file into a Chime Pro query form box. The function should conform to the prototype:

```
function JSFunctionName( pluginName )
```

where *pluginName* is the **name** specified for the Chime structure window.

Syntax

LoadStructCallback={JSFunctionName}

Note that if you are using frames you should explicitly specify the document containing the function, for example:

```
LoadStructCallback=parent.frameName.MyLoadStructCallback
```

Example

```
<script>
function MyLoadStructCallback( plugname ) {
  // add your code here ...
}
</script>
<embed type='chemical/x-mdl-molfile' name="themol"
width=270 height=170 bgcolor=white, display2d=true
queryformbox='document.qbuild.csFld_molstructure'
LoadStructCallback="MyLoadStructCallback"
>
```

MessageCallback

Specifies a JavaScript function in the document that is to be called whenever an informational message is generated by Chime's 3D rendering processes. The function should conform to the prototype:

```
function JSFunctionName(pluginName, messageText)
```

where *pluginName* is the **name** specified for the Chime structure window and *messageText* is the informational message from Chime. Note that if you are using frames you should explicitly specify the document containing the function, i.e., specify:

```
MessageCallback=parent.frameName.MyCallBack
```

Syntax

MessageCallback={JSFunctionName}

See also

messages3d

Example

```
<script>
function MyMessageCallback( plugname, message ) {
  // add this message to our message textarea....
  document.theform.chimemessages.value += message + "\r\n";
}
</script>
<embed src="asprin.pdb" name="themol" width=400 height=350
MessageCallback="MyMessageCallback"
>
<form name="theform">
Chime Messages Output:<br>
<textarea name="chimemessages" rows=20 cols=80></textarea>
</form>
```

messages3d

If `true`, messages generated by normal operation of Chime's 3D rendering engine are echoed to the browser's status line.

Syntax

```
messages3d={false|no|true|yes}
```

false or **no** - do not echo messages

true or **yes** - echo messages to status line

See also

MessageCallback

---

name

Specifies the name of the plug-in. This is needed to make a Chime the **target** of a Chime button. It is also used in JavaScript when executing a script through Chime 2.0's LiveConnect interface. Both techniques are shown in the example below.

Syntax

```
name={name}
```

See also

button
ButtonCallback
target

Example

```
<!-- function starts the mol spinning using LiveConnect interface -->
<script language="JavaScript">
function SpinMol(pluginName, executedYet) {
  if (!executedYet) { //make sure only executes once per click
    document.themol.executeScript("spin");
  }
}
</script>
<!-- the first Chime displays the structure -->
<embed src="apa.pdb" width=400 height=350
spiny=80 startspin=false
name="themol"
><br>
<!-- this button executes a script directly using script tag -->
<embed type="application/x-spt" width=30 height=30
button=push script="spin" target=themol
>
<!-- this button does the same thing through the JS function -->
<embed type="application/x-spt" width=30 height=30
button=push ButtonCallback=SpinMol
>
```

---

nmrpdb

Sets the MIME type to the NMR pdb mime type so that "multi-frame" pdb files can be loaded.

Syntax

nmrpdb={false|true|auto}

false – use default PDB file reading, even if there are MODEL records present

true - treat the PDB file as an NMR model and ignore all CONECT records

auto – treat the PDB file as an NMR model if a MODEL record is found

---

nomenus

Specifies whether to display Chime menus. This feature allows you to turn off Chime's menus except the About menu.

Syntax

nomenus={true|false}

true – do not display Chime menus except for the About menu

false – display all Chime menus

---

options3d

Specifies 3D display options. If more than one options3d option is specified for the EMBED tag the options will be combined.

Syntax

options3d={dots|hetero|hydrogen|labels|shadows|slab|specular|stereo}

The options correspond to those available in the Chime menu.

See also

color3d
display3d
scale3d

Example

```
<!-- this embed has two 3d options -->
<embed src="apa.pdb" width=400 height=350
display3d=spacefill options3d=dots options3d=labels
>
```

---

palette

"Foreground" allows Chime to use the colors it needs outside of the current color palette to smoothly display spacefilling structures. This corresponds to the "Force Palette" command on the Chime menu.

Syntax

palette={background|foreground}

See also

bgcolor
color3d

PauseCallback

Specifies the name of a JavaScript function in the current document that is to be called whenever a `pause` command is encountered in a RasMol script. The function is called twice: once when the `pause` is first encountered and again when the script resumes. The function must conform to the prototype:

```
function JSFunctionName( pluginName, begPause? )
```

...where *`pluginName`* is the **`name`** of the Chime in which the script is running, and the boolean *`begPause?`* is `true` when the `pause` is first encountered and `false` when execution of the script resumes.

Syntax

`PauseCallback={JSFunctionName}`

PeakCallback

Specifies the name of a JavaScript function in the current document that is to be called whenever the user clicks on an peak in a JCAMP-DX spectrum window. The function should conform to the prototype

```
function JSFunctionName( pluginName, peakExpression )
```

...where *`pluginName`* is the **`name`** of the Chime structure window and *`peakExpression`* is a string representing the X,Y coordinates of the position on which the user has clicked.

Syntax

`PeakCallback={JSFunctionName}`

Example

```
<script>
function MyPeakCallback( plugname, message ) {
  // add this message to our message textarea....
  document.theform.chimemessages.value += message + "\r\n";
}
</script>
<embed src="pyridine.jdx" name="theNMR" width=400 height=350
jcamp_grid="true" jcamp_revplot="true"
MessageCallback="MyPeakCallback"
>
<form name="theform">
Chime Messages Output:<br>
<textarea name="chimemessages" rows=20 cols=80></textarea>
</form>
```

See also

jcamp_blocknum
jcamp_grid
jcamp_revplot
jcamp_help

PickCallback

Specifies the name of a JavaScript function in the current document that is to be called whenever the user clicks on an atom in a Chime structure window. The function should conform to the prototype

```
function JSFunctionName( pluginName, atomExpression )
```

...where *pluginName* is the **name** of the Chime structure window and *atomExpression* is a string representing the atom (including residue information for a pdb file) on which the user has clicked.

Syntax

PickCallback={JSFunctionName}

Example

```
<script>
function MyPickCallback( plugname, message ) {
  // add this message to our message textarea....
  document.theform.chimemessages.value += message + "\r\n";
}
</script>
<embed src="3dfr.pdb" name="themol" width=400 height=350
display3d=ball&stick
MessageCallback="MyPickCallback"
>
<form name="theform">
Chime Messages Output:<br>
<textarea name="chimemessages" rows=20 cols=80></textarea>
</form>
```

---

preloadscript

Specifies a RasMol script to be executed immediately when the plugin is loaded, **before** any file is loaded from the **src** tag. Multiple commands can be separated with a "|" or ";".

Syntax

preloadscript={RasMol script commands}

See also

immediate
script

reactingcenters2d

Reacting centers indicate where specific transformations occur on a reaction. Applies to reactions in 2D.

Syntax

`reactingcenters2d={off|color|thicken|hash}`

`off` – do not display reacting centers

`color` - display reacting centers in color

`thicken` - display reacting centers thickened

`hash` - display reacting centers with hashes

See also

atommapnum2d
invretmarks2d

---

scale3d

Specifies a scaling value (in Angstroms per inch) to be used when a 3D structure is displayed. When the same **`scale3d`** value is used for multiple Chime structure windows on a page, the viewer can get a feel for the relative size of the structures, as shown in the example.

Syntax

scale3d={real value}

The value should be at least 10 angstroms/inch. The higher the scaling value, the smaller the molecule will appear.

See also

color3d
display3d
options3d
scale3d

Example

```
<!-- this displays two structures at same scale -->
<embed src="3dfr.pdb" width=400 height=350
display3d=ball&stick scale3d=10
>
<embed src="apa.pdb" width=400 height=350
display3d=ball&stick scale3d=10
>
```

---

script

Specifies any valid RasMol script commands to apply to the plug-in. Multiple commands can be separated with a "|" or ";".

When attached to a Chime structure window, the script executes immediately when the plug-in loads. When attached to a Chime button (**button**=true), the script executes in the Chime structure window indicated by the **target** tag when the button is pushed.

Syntax

script={valid RasMol script commands}

See also

altscript
csml
debugscript
immediate
preloadscript
target

---

sendmouse

Specifies whether to send Chime's mouse events to another target Chime plug-in.

Syntax

sendmouse={false|true}

false – do not send mouse events to another target Chime plug-in
ftrue – send mouse events to another target Chime plug-in

Example

In the following example each plug-in points to the next one, and the last one (`chime3`) points back to the `chime1` plug-in. By doing this you can link 2 or more plug-ins mouse events.

```
<embed name=chime1 src="connect_test.pdb" width=45% height=70%
initscript="set connect save"  target=chime2 sendmouse=true
>
<embed name=chime2 src="connect_test.pdb" width=45% height=70%
```

```
initscript="set connect false"  target=chime3 sendmouse=true
>
<embed name=chime3 src="connect_test.pdb" width=45% height=70%
target=chime1 sendmouse=true
>
```

---

spinfps

Specifies the frame rate per second for spinning. The default is 10.

Note: The value specified for **spinfps** may not be possible on the viewer's platform. Chime will get as close as the system allows.

Syntax

spinfps={spin frame rate / second}

See also

spinX
spinY
spinZ
startspin

Example

```
<!-- spins along every axis, 100 frames per second -->
<embed src="apa.pdb" width=400 height=400
spinX=20 spinY=50 spinZ=80 spinfps=100 startspin=true
>
```

---

spinX

Specifies the rotation speed along the X-axis (horizontal axis) in degrees per second. The default is 0 (no spin).

Syntax

```
spinX={degrees per second}
```

See also

spinfps
spinY
spinZ
startspin

Example

```
<!-- spins along every axis, 100 frames per second -->
<embed src="apa.pdb" width=400 height=400
spinX=20 spinY=50 spinZ=80 spinfps=100 startspin=true
>
```

---

spinY

Specifies the rotation speed along the Y-axis (vertical axis) in degrees per second. The default is 30.

Syntax

spinY={degrees per second}

See also

spinfps
spinX
spinZ
startspin

Example

```
<!-- spins along every axis, 100 frames per second -->
<embed src="apa.pdb" width=400 height=400
spinX=20 spinY=50 spinZ=80 spinfps=100 startspin=true
>
```

---

spinZ

Specifies the rotation speed along the Z-axis in degrees per second. This is the axis that "comes out of the screen" toward the user. The default is 0 (no spin).

Syntax

spinZ={degrees per second}

See also

spinfps
spinX
spinY

startspin

Example

```
<!-- spins along every axis, 100 frames per second -->
<embed src="apa.pdb" width=400 height=400
spinX=20 spinY=50 spinZ=80 spinfps=100 startspin=true
>
```

---

src

Specifies what file to open and display. This can be a MOL (MDL molecule) file, a PDB (Protein Databank) file, or an XYZ animation file.

In general, if you use the **src** tag, you do not need to also specify an embed **type**. The embed type is determined automatically.

Syntax

src={file name}

See also

structure

---

ssbonds

Specifies whether, and how, to display disulphide bonds.

Syntax

ssbonds={off|on|number}

off - do not display disulphide bonds

on - display disulphide bonds as dashed lines

number - display disulphide bonds as cylinders whose diameter is proportional to the number.

See also

hbonds

---

startanim

Set to `true` if you want an animation in a concatenated XYZ file to begin immediately. If `false`, you can start the animation using the RasMol script command `"animation on"`.

Syntax

startanim={false|no|true|yes}

See also

animfps
AnimFrameCallback
animmode

Example

```
<embed src="sn2.xyz" display3d=ball&stick width=300 height=200
animfps=30 startanim=true animmode=loop
>
```

---

startspin

Set to `true` if you want the structure to start spinning immediately upon loading. If `false`, you can start the spin with the RasMol script command `"spin"`.

Syntax

startspin={true|yes|false|no}

See also

spinfps
spinX
spinY
spinZ

Example

```
<!-- spins along every axis, 100 frames per second -->
<embed src="apa.pdb" width=400 height=400
spinX=20 spinY=50 spinZ=80 spinfps=100 startspin=true
>
```

---

structure

Used instead of **src** to load an in-line compressed molfile. The compressed molfile is a compressed, URL-safe encoded version of the structure file. It is generated by Chemscape Server. You must specify the mime type of the plug-in via the **type=** tag.

Syntax

```
structure={compressed molfile}
```

See also

src
type

---

target

In a Chime button, specifies which Chime structure window should execute the attached RasMol or CSML script. The other Chime is referenced according to its **name**.

Syntax

```
target={target Chime name}
```

See also

button
csml
immediate
name
script

type

Used to specify the mime type of the plug-in when the **structure=** option is used.

Syntax

```
type={Chime_mime_type}
```

See also

structure

Example

---

version

Specifies the release version of MDL Chime

Syntax

```
script="show version"
```

Example

```
<script>
function MyMessageCallback( sPluginName, message )
{
        if (message.indexOf("2.6 SP5")>=0)
```

```
        {
             alert("Version 2.6 SP5. You have the latest version");
        }
        else if (message.indexOf("1.0")>=0)
        {
          alert("For this example to work, you need to install MDL Chime 2.6 SP5");
          windowOpener();
        }
}
</script>

<embed src="test.mol" width='150' height='400' bgcolor='white' name='MOLdisplay'
 messagecallback='MyMessageCallback' script='show version;zoom 150' \></embed>
```