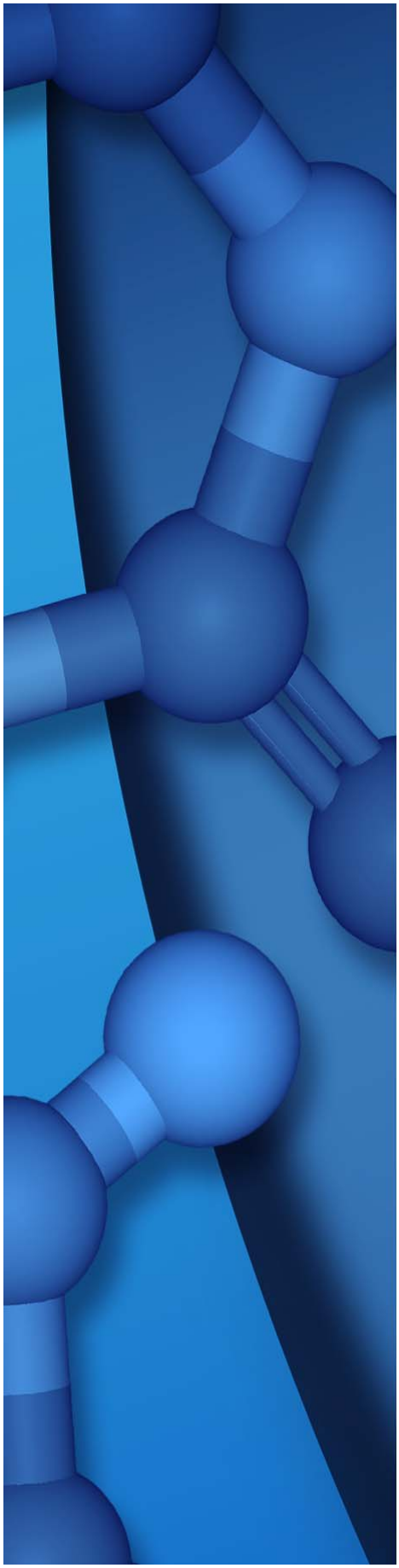


accelrys®

ACCELRYS DRAW 4.2

Developer's Guide

Last updated on: 29 July 2013

July 2013

Acknowledgments and References

Accelrys may grant permission to republish or reprint its copyrighted materials. Requests should be submitted to Accelrys Scientific and Technical Support, either through electronic mail to , or in writing to:

Accelrys Scientific and Technical Support
10188 Telesis Court, Suite 100, San Diego, CA 92121

Accelrys Draw Overview

About Accelrys Draw

Accelrys Draw is an application for drawing, displaying, and editing chemical structures, including biopolymers. Accelrys Draw has a similar look-and-feel as ISIS/Draw, but offers new capabilities. The application developer can: use the Draw API to create custom actions and add-ins that extend functionality for tools, buttons, and menu items; create Isentris® applications that call Draw; and add Draw components, such as the Renderer and Renditor, into Microsoft® Visual Studio® .NET applications. In addition, the XML configuration file can be edited to make the chemical drawing environment match workflow needs.

Note: The Accelrys JDraw Applet for drawing and rendering in web applications is available at no charge to Accelrys Draw Enterprise Edition licensees as well as to academic and non-profit institutions. See `http://accelrys.com/products/informatics/cheminformatics/draw/`

Customizing Accelrys Draw

You can customize Accelrys Draw in three ways:

- Changing the User Interface: Edit the XML Configuration File
- Cheshire Actions and Custom Add-ins
- Using Draw in a .NET application

Accelrys Draw includes example applications that involve the API as well as examples that involve the XML configuration file only. See Examples Summary.

Note: Accelrys Draw also allows you to work with biopolymers. See the Biopolymer Representation chapter in *Accelrys Chemical Representation*, which is installed as `<draw_home>\docs\Draw\devguide\AccelrysChemicalRepresentation.pdf`

Changing the User Interface: Edit the XML Configuration File

The Accelrys Draw administrator can customize the graphical user interface of Accelrys Draw by editing the XML configuration files. The default name of the primary XML configuration file is `AccelrysDraw-net.xml`. By default, `AccelrysDraw-net.xml` resides in the root installation folder that also has `AccelrysDraw.exe`, the product's executable.

For examples of editing the XML configuration files, see the Accelrys Draw Configuration Guide chapter named "Accelrys Draw Configuration Tasks".

Note: The XML configuration file affects the initial loading of Accelrys Draw. If you want to change Accelrys Draw behavior at runtime, use the API. See Using Draw in a .NET application.

Cheshire Actions and Custom Add-ins

Accelrys Draw includes examples of how to extend Accelrys Draw functionality.

Feature	Description
Cheshire Action	Allows incorporating Accelrys Cheshire scripts as "actions" in Accelrys Draw. See Cheshire Actions
Add-in Action (or tool)	Allows incorporating third-party functionality. See Add-ins: Actions and Tools

Note:

- In addition, you can specify which set of atom types (or Periodic table) Accelrys Draw uses. See Setting a Custom Ptable.
- For a comprehensive discussion of the Accelrys Ptable, see the Customizing the Accelrys Ptablechapter in *Accelrys Chemical Representation*, which is installed as `<draw_home>\docs\Draw\devguide\AccelrysChemicalRepresentation.pdf`

Using Draw in a .NET application

The Accelrys Draw API provides a standard .NET interface that you can call from your application. See API: Getting Started and API References - Links.

You can use the Accelrys Draw API to perform tasks such as the following.

- Use Accelrys Draw in a .NET application. For example, including the Renderer or Renditor in a .NET form.
- Use a Renderer and the Renditor in an HTML page. A web server application can accept query structures from web clients and return matching structures to those web clients. Both the web server and the clients must have an installation of Accelrys Draw for .NET.
- Extend the functionality of the structure Editor's menu item "actions" and toolbar button "tools" by using:
    - Your own Cheshire scripts.
    - Add-ins for third-party functionality
    - Custom Add-ins for functionality your write yourself. Interfaces exist to the structure Editor as well as to the selected molecule, atom, bond, Sgroup, menu item, or tool.
- Generate multiple image files (bmp, gif, png, tif, wmf) from structure data for reporting by using the HeadlessRenderer's `SaveMoleculeAsImage` method. (Compare to the Accelrys Draw Configuration Guide section on "Command-line Image File Conversion: ImageGenerator".)
- Create a common look and feel for the display of structures. For example, you can ensure that when a renditor calls an editor, the renditor's DisplayPreferences (such as `BackColor` and `AromaticRingsAsCircles`) apply to the editor.

Using Draw in a Java application

A Java application can launch the Accelrys Draw .NET renderer and editor by using the API of `com.mdli.draw.JMolRendererCom` at `<draw_home>\docs\Draw\api-java\index.html`

The API Reference is available at `JMOLRendererCOM` javadoc

Components, Classes, and Utilities of Accelrys Draw

Accelrys Draw provides components for drawing, displaying, and editing chemical structures (molecules and reactions), as well as a command-line utility for image file generation.

To customize Accelrys Draw for your application, use the following components, classes, and tools.

To learn how to configure Accelrys Draw by editing XML configuration files, see the Accelrys Configuration Guide chapter on “Accelrys Draw Configuration Tasks”.

Component	Description	For More Info, see Accelrys Draw API Reference for .NET
`structureEditor`	enables the end-user to draw and modify chemical structures. The Editor is also the Accelrys Draw standalone application.	`IEditor` interface
`Renderer`	a control that displays structures	`Renderer` class
Renditor	a special renderer control that enables the end-user to launch a structure Editor by double-clicking anywhere on the canvas	`Renditor` class
`HeadlessRenderer`	an invisible renderer control that paints a structure to an image file. Note: A command-line utility also outputs image files (see the Accelrys Draw Configuration Guide section on “Command-line Image File Conversion: ImageGenerator”)	`HeadlessRenderer` class

Two classes are particularly important.

Class	Description	For More Info, see Accelrys Draw API Reference for .NET
`DisplayPreferences`	encapsulates the settings of display settings (such as background color and whether aromatic rings appears as circles or alternating double bonds)	`DisplayPreferences` class
`StructureConverter`	converts structure files from one format to another. For example, you can generate molfiles from Chime and SMILES.	`StructureConverter` class

A command line utility:

Command-line Utility	Description	For More Info, See ...
`ImageGenerator`	outputs an image file that corresponds to the structure file input	`ImageGenerator` example (see the Accelrys Draw Configuration Guide section on "Command-line Image File Conversion: ImageGenerator"))

Related topics

- Components, Classes, and Utilities of Accelrys Draw
- Changing the User Interface: Edit the XML Configuration File
- For More Information About Accelrys Draw

For More Information About Accelrys Draw

Accelrys Draw component, class, or task:	see Accelrys Draw Developer Documentation:
`Renderer`	Developer's Guide > Accelrys Draw .NET API: Getting Started Draw .NET API Reference > MDL.Draw.Renderer.Renderer
`Renditor`	Developer's Guide > Accelrys Draw .NET API: Getting Started Draw .NET API Reference > MDL.Draw.Renditor.Renditor
`HeadlessRenderer`	Draw API Reference > MDL.Draw.HeadlessRenderer See also the "Demo Headless Renderer" example in the Examples folder of your Draw installation
`DisplayPreferences`, including how to make multiple renditors synchronize their display settings	Developer's Guide > .NET API: Getting Started Contents > DemoRenditorMulti Example - a grid of renditors and sharing display preferences Draw API Reference > MDL.Draw.Renderer.Preferences
Editor and extending functionality	Developer's Guide > Add-ins: Actions and Tools Draw .NET API Reference > MDL.Draw.Interfaces.IEditor
Using Accelrys Cheshire in Accelrys Draw	Developer's Guide > Cheshire Actions in Accelrys Draw .NET
Using Accelrys Draw in Internet Explorer or an ActiveX control	Developer's Guide > Examples: Internet Explorer and C++
Accessing Accelrys Draw .NET from a Java application	The javadoc that is, by default, installed at `<draw_home>\docs\Draw\api-java\index.html`

Note: See API References - Links.

Examples Summary

Accelrys Draw examples are installed in the Examples directory under the root directory of the installation (which this documentation refers to as <DRAW_HOME>). The following examples are described and illustrated in this Developer's Guide.

Example Location: <DRAW_HOME>\Examples\	**Description**
ASP\	Shows how the server with the .asp pages responds to the client. ASP Example
C#\AddInActionExample\	An Add-in action that displays a SMILES string that represents the current molecule. See About Add-in Actions.
C#\DemoExtendableRenditor	How to make the renditor launch a customized Editor.
C#\DemoHeadlessRenderer\	The Render Mode menu lets you paint the structure to Graphics, Bitmap, or Metafile. See DemoHeadlessRenderer Example.
C#\DemoLateBoundRenditor	How to launch the ClickOnce version of Draw if the MSI Version is not found.
C#\DemoRenderer\	To quickly see what various API settings do. See DemoRenderer Example.
C#\DemoRendererCustomAssemblyResolver	How an application can target either the ClickOnce or MSI version of Draw.
C#\DemoRenditorMulti\	Shows four special Renderers you can click on to get an Editor. Explains how to synchronize display settings among multiple renditors. See DemoRenditorMulti Example - a grid of renditors and sharing display preferences.
C#\DemoRenditorSimple\	Shows one Renditor and the molfile that corresponds to the current structure. See DemoRenditorSimple Example.
C#\DemoStructureResolver	Show how to create a custom structure resolver.
C#\FacadeAPIActionExample	Shows how to programmatically manipulate atoms, bonds, and sketch objects, such as circles, lines, polygons, and text.
C#\GettingStarted\	Shows how to build a .NET application that displays and converts structures. See .NET API GettingStarted Example.
C++\LateBindingCLIRenderer C++\Renderer_MFC C++\Renditor_MFC	Shows how C++ applications can use Draw .NET controls. See C++ Examples.
CheshireAction\	Shows how to use a Accelrys Cheshire script. See the Cheshire Actions chapter in the Configuration Guide.
IE\	Shows how Internet Explorer can host Draw .NET controls. See Examples: Internet Explorer and C++.
Java\	Java Native Interface (JNI) Wrapper for accessing Draw .NET from a Java application. See Java Native Interface - JNI Wrapper.

Example Location: <DRAW_HOME>\Examples\	**Description**
Ptable\	Shows how to make Draw use a customized periodic table. See Setting a Custom Ptable and Specifying the Custom Ptable and Custom Weights.
SequenceTool\	(Biopolymer support) - See the API Reference for StructureConverter and Renderer , as well as the Biopolymer Representation chapter in *Accelrys Chemical Representation*, which is installed as <draw_home>\docs\Draw\devguide\AccelrysChemicalRepresentation.pdf
VB.Net\AddInToolExample-VB.Net\	Identifies which atom or bond the end-user selected.
VB.Net\DemoHeadlessRenderer\	The Render Mode menu lets you paint the structure to Graphics, Bitmap, or Metafile. See DemoHeadlessRenderer Example
VB.Net\DemoRenderer\	To quickly see what various API settings do. See DemoRenderer Example
VB.Net\DemoRenditorMulti\	Shows four special Renderers you can click on to get an Editor. See DemoRenditorMulti Example - a grid of renditors and sharing display preferences
VB.Net\DemoRenditorSimple	Shows one Renditor and the molfile that corresponds to the current structure. See DemoRenditorSimple Example
VB.Net\GettingStarted\	Shows how to build a .NET application that displays and converts structures. See .NET API GettingStarted Example

Note:

- To access the C# examples, use <DRAW_HOME>\Examples\C#\csharpexamples.sln
- To access the VB examples, use <DRAW_HOME>\Examples\VB.Net\VB.NetExamples.sln
- To access the Java example, go to <DRAW_HOME>\Examples\Java

Important Files

The following are important files that you use when modifying or running Accelrys Draw. They are located in the root directory of the Accelrys Draw installation.

File Name	Description
`AccelrysDraw.exe`	Draw executable (also available from the Start menu)
`AccelrysDraw-net.xml`	XML configuration file that specifies the XML files that define the Draw user interface. For reference details, see the Accelrys Draw Configuration Guide section named "About the Accelrys Draw XML Configuration Files".

Note: If you have a custom Ptable (periodic table), see the Customizing the Accelrys Ptablechapter in *Accelrys Chemical Representation*, which is installed as `<draw_home>\docs\Draw\devguide\AccelrysChemicalRepresentation.pdf`

A convenient way to access Accelrys Draw examples from within a Microsoft Visual Studio .NET environment is to launch the `Examples.sln` solution file in the `Examples` directory of the Accelrys Draw installation. For an overview of how to build you own application that customizes Accelrys Draw, see API: Getting Started.

Note: The structure files for your corporate templates are important. Accelrys recommends that you place them in a read-only directory. See the Accelrys Draw Configuration Guide section named "Templates: Customizing".

About the Accelrys Draw Editor

The Editor control makes it possible to create and modify structures. The Accelrys Draw API does not provide direct access to the Editor.

An end-user can interact with the Editor by launching Draw.exe or by double-clicking a Renditor, which then launches the Editor.

Note:

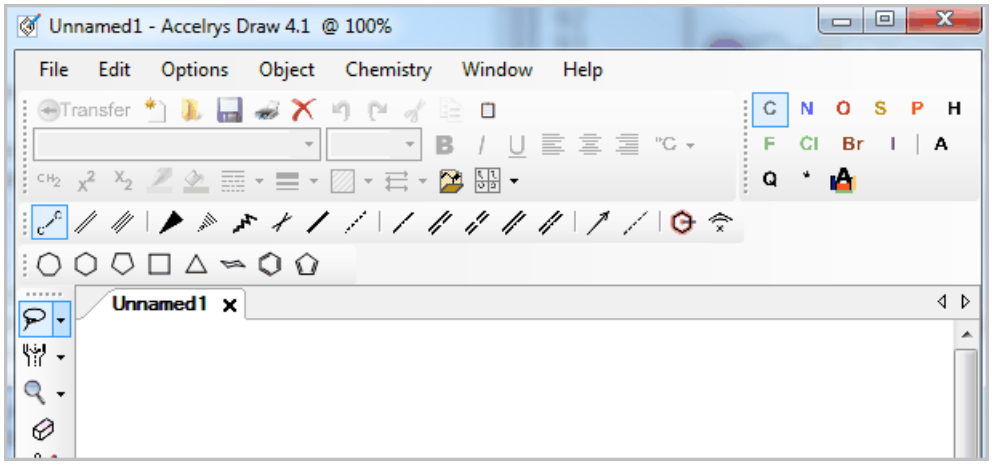
- For an example of how an application that adds functionality to Accelrys Draw can get a handle to the Editor by using the IEditor interface, see Add-ins: Actions and Tools. For details and syntax, see *Draw API Reference for .NET*.
- Draw, by default, is a modeless application. There is no API to make the Draw editor modal. However, if the form that contains the Renditor that invokes the editor is:
  - **modal**, then the editor will also be **modal**
  - **modeless**, then the editor will also be **modeless**

Starting Accelrys Draw

When a user starts Accelrys Draw, it is ready for the end-user to display, create, and edit structures.

See also: Specifying XML Configuration Files.

To start Accelrys Draw, run the `AccelrysDraw.exe` executable, which resides the root installation folder.



API: Getting Started

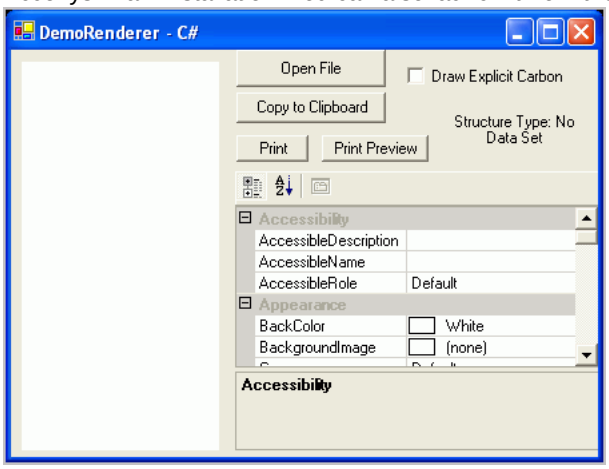
About the Accelrys Draw API

Accelrys Draw provides an application programming interface (API) for .NET applications. This API provides access to these components (see Examples Summary). In addition, there are interfaces that support Add-in actions and tools. For details and syntax, see *Draw API Reference for .NET*, available from API References - Links.

Note: Accelrys Draw includes example applications that involve the API as well as instructions that involve the XML configuration file only. See Examples Summary and .NET API GettingStarted Example.

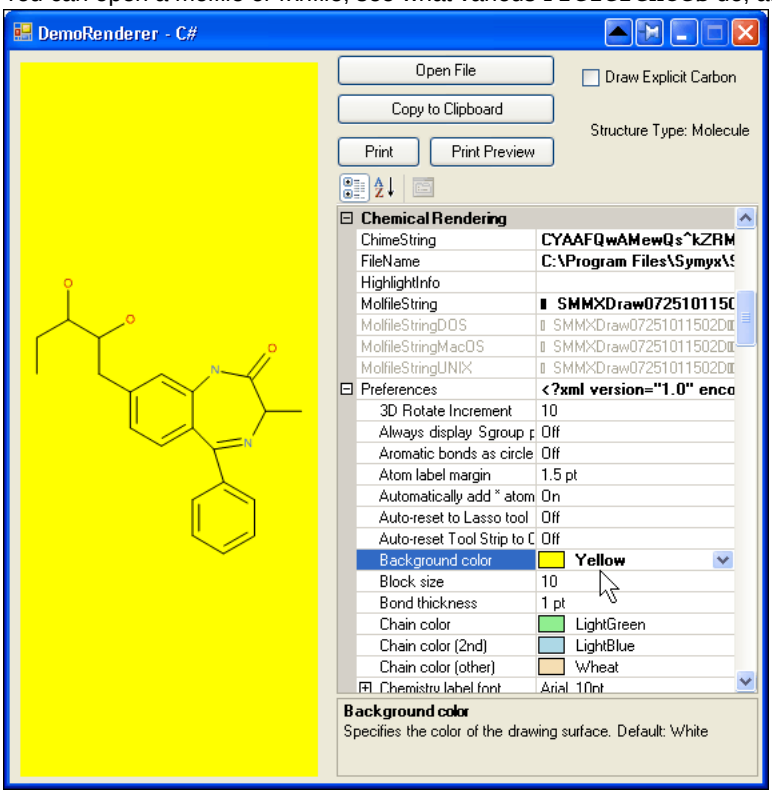
DemoRenderer Example

To quickly see what various API settings do, experiment with the DemoRenderer example. `DemoRenderer-csharp.exe` is located in the `Examples\C#\DemoRenderer\bin\Release\` subfolder of your Accelrys Draw installation. You can also launch it from the Start menu.



Note: There is also `DemoRenderer-VB.Net.exe in the \Examples\VB.Net\DemoRenderer\bin` subfolder of your Accelrys Draw installation. You can also launch it from the Start menu.

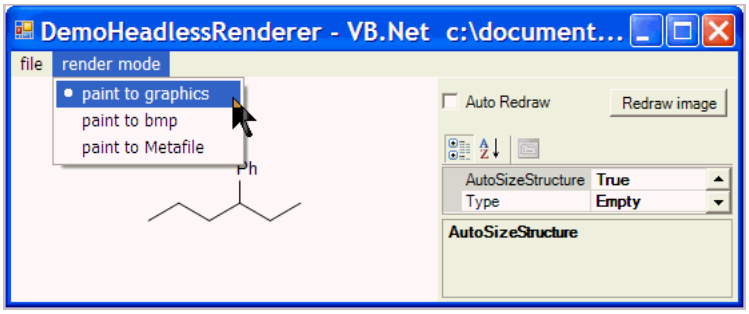
You can open a molfile or rxnfile, see what various **`Preferences`** do, and also set other properties.



For an overview of other examples, see Examples Summary.

DemoHeadlessRenderer Example

The DemoHeadlessRenderer example enables you to load a structure from the File menu, and then call the overloaded PaintMolecule method by using the render mode menu:



For an overview of other examples, see Examples Summary.

.NET API GettingStarted Example

This walkthrough of the GettingStarted example shows how to use the Draw API in the Microsoft Development Environment for .NET. This example uses Visual Basic .NET (see the `\Examples\VB.Net` directory).

Note: For the C# version, see the `\Examples\C#` directory .

Follow these steps to create a sample application that:

- Calls the `StructureConverter` component to convert a Chime string to a molfile string and a molfile string to a Chime string
- Reads in a molfile and a SMILES file, and calls the `Renderer` control to display the structure

In this walkthrough, we will:

- Design the Form
- Code the behavior for structure conversion and molecule rendering
- Test the application

Note: The .NET solution file, `Examples.sln`, in the `Examples` directory of your Accelrys Draw installation, allows you to work with multiple examples. To work with just one project, such as `GettingStarted`, on the **`View`** menu, select **`Solution Explorer`**, right-click on the project and select **`Set As Startup Project`**.

Designing the Form

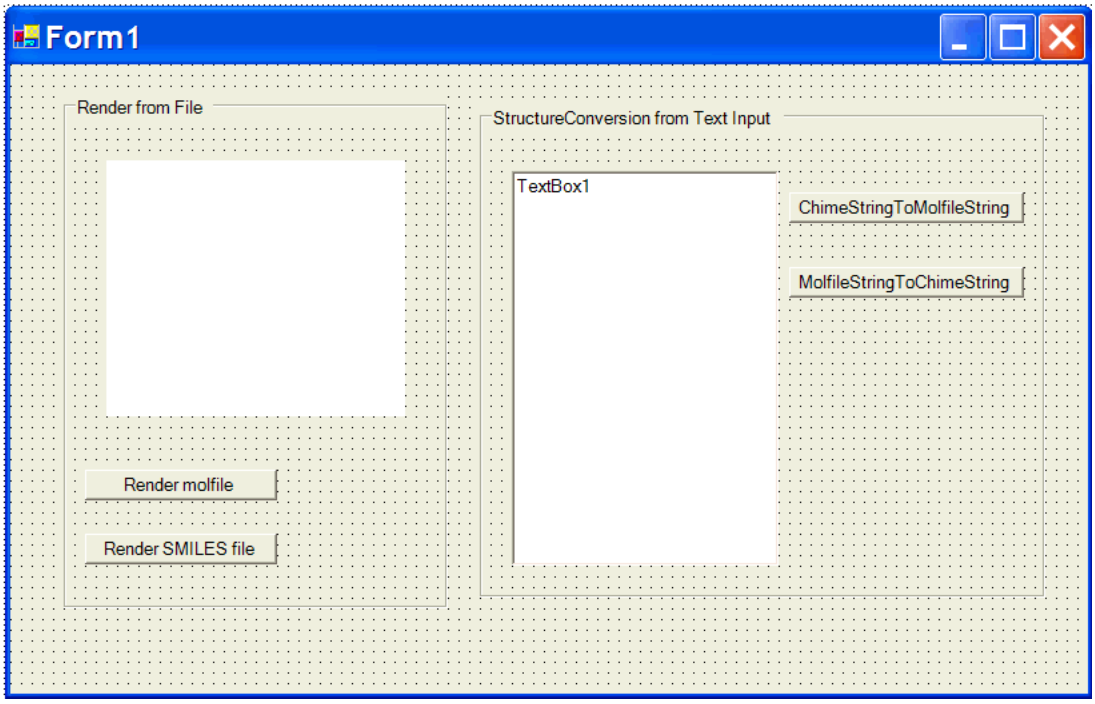
1. Create a directory in which to put your solution and project.
2. Open your development environment. Create a new Visual Basic project using the Windows Application template. Fill in the following information:
   - Name: `ConverterRenderer`
   - Location: The directory you created
   - New Solution Name: `ConvertAndRender`
3. Copy `test.mol` and `test.smi` from the `Examples\VB.Net\GettingStarted` directory of the Accelrys Draw installation to the `ConverterRenderer` directory.
4. On the Project menu, click Add Reference, browse to the `\lib` directory of your Accelrys Draw installation and select these DLLs.
   - `MDL.Draw.Foundation.dll`
   - `MDL.Draw.Renderer.dll`
5. Add the `Renderer` control to the Toolbox:
   - Open the Toolbox. Right-click in it and choose `Add/Remove Item...`
   - Click the Browse button, navigate to the `\lib` directory of your Accelrys Draw installation, and select `MDL.Draw.Renderer.dll`. A Renderer control now should appear in the Toolbox. (We will not use the `HeadlessRenderer` control which was also added to the Toolbox.)
6. Add a GroupBox to the form and set its Text to "Render from File"
7. Add a `Renderer` control to the "Render from File" GroupBox
8. Add two buttons below the `Renderer` conrol in the "Render from File" GroupBox with the following settings:

Button Name	Button Text
Button1	Render molfile
Button2	Render SMILES file

9. Add a second GroupBox to the right of the first GroupBox. Set its Text to "StructureConversion from Text Input"
10. Add a TextBox control to the "StructureConversion from Text Input" GroupBox. Set `AutoSize` to `False` and `Multiline` to `True`.
11. Add two buttons to the right of the TextBox in the "StructureConversion from Text Input" GroupBox. Use the following settings:

Button Name	Button Text	Enabled
Button3	ChimeStringToMolfileString	True
Button4	MolfileStringToChimeString	False

12. Edit the Form to look like the one shown below:



Continue to the next section on Coding the Behavior.

Coding the Behavior

1. Set the Text property of the TextBox to the following Chimestring by pasting the these characters (do not try to type this long string of characters):

   ```
   tYg7bj3AEt7QWOqIh387jIU2AukGT0r621Cmplzmp5iHirhwnG1X6dLRGEG6kKqpaFVRXe29E2fo1YILX
   34JPltQ9icr^vQv5$sTFkFlWbwSyFpZWgfoXIyJXdrA9PZRr6d8Aow9SxniNNlMMLz^v0lqUHbzz^mLEX
   le3JRA1dZtl4Ym1pVrZUi1EVmqeOMqGP$gmvTQZIBUqsATSAYKrBXitYDQAwFMGHXpObghbT5ul4wPThO
   1$bP73qQN44O45JQRVV9P5uvyv1xSsA9hdcxcGTggoUir8midJ1libImabMlmOd0CUxaXpq0cHTEtX$i^
   bzecmjVdlB^ApHe4HiRd29QtP$srpIeQMCOdS3Z90^v13ucCLL9qB83l7FW6mC
   ```

2. Double-click the `ChimeStringToMolfileString` button and add the following code to the `Button3_Click` event.

   ```
           Button4.Enabled = True  ' Enable MolfileStringToChimeString
           Button3.Enabled = False ' Disable ChimeStringToMolfileString
           ' Display MolfileString when Button3 is clicked
           TextBox1.Text =
   MDL.Draw.StructureConversion.StructureConverter.ChimeStringToMolfileString(TextBo
   x1.Text)
   ```

   The preceding code calls the Accelrys Draw `StructureConverter` to convert the Chimestring to a molfile string. It also disables the `ChimeStringToMolfileString` button and enables the `MolfileStringToChimestring` button.

3. Double-click the `MolfileStringToChimestring` button and add the following code to the `Button4_Click` event.

   ```
           Button3.Enabled = True   ' Enable ChimeStringToMolfileString
           Button4.Enabled = False  ' Disable MolfileStringToChimeString
           ' Display ChimeString when Button4 is clicked
           TextBox1.Text =
   MDL.Draw.StructureConversion.StructureConverter.MolfileStringToChimeString(TextBo
   x1.Text)
   ```

   The preceding code calls the Accelrys Draw `StructureConverter` to convert the molfile string to a Chimestring. It also disables the `MolfileStringToChimestring` button and enables the `ChimeStringToMolfileString` button.

4. Double-click the `Render molfile` button and add the following code to the `Button1_Click` event.

   ```
           Dim sr As System.IO.StreamReader = New
   System.IO.StreamReader("../test.mol")
           Dim newMolfileString As String
           newMolfileString = sr.ReadToEnd()
           ' Change the HighlightColor property to Red
           Renderer1.Preferences.HighlightColor = System.Drawing.Color.Red
           Renderer1.MolfileString = newMolfileString
   ```

   The preceding code reads in a molfile (test.mol), sets the highlight color to red, and calls the `Renderer` to display that structure.

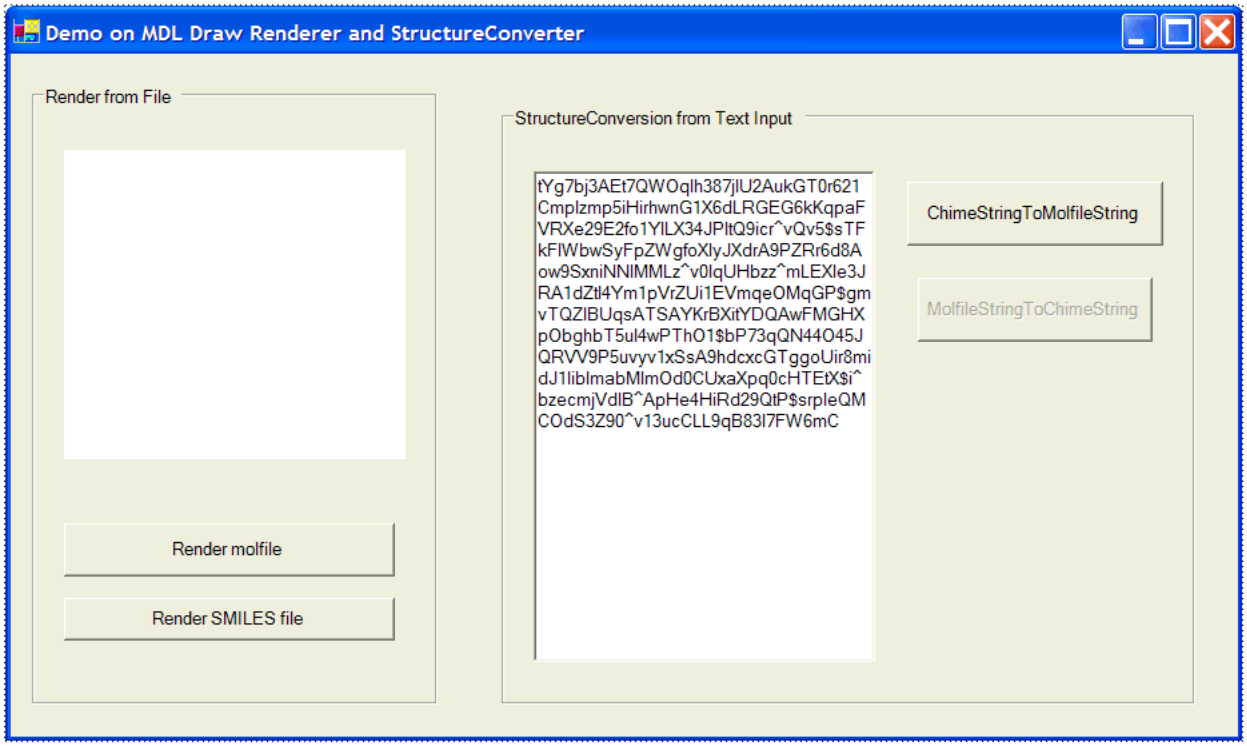
5. Double-click the `Render SMILES file` button and add the following code to the `Button2_Click` event.

```
        Dim sr As System.IO.StreamReader = New
System.IO.StreamReader("../test.smi")
        Dim newSmilesString As String
        newSmilesString = sr.ReadToEnd()
        Renderer1.SmilesString = newSmilesString
```

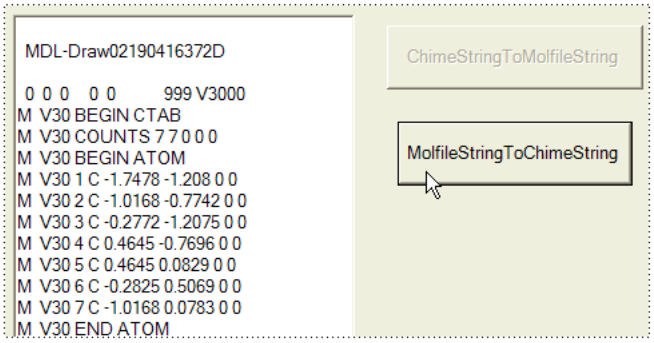
   The preceding code reads in a SMILES file (test.smi) and calls the `Renderer` to display that structure.

Testing the Application

1. Build and run the project. The application appears with a Chimestring in the text box.

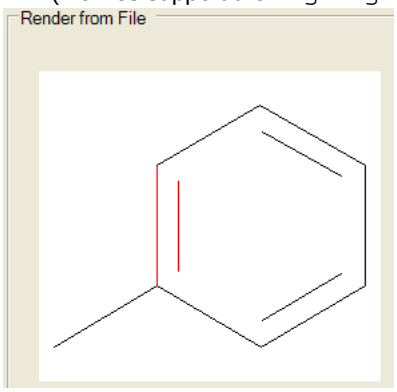


2. Click the `ChimeStringToMolfileString` button and the molfile string replaces the Chimestring in the TextBox.

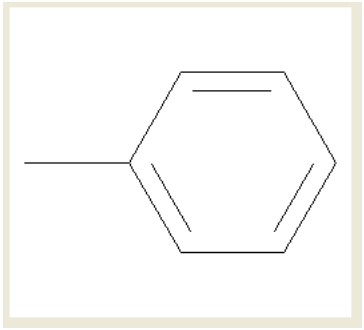


3. Click the `MolfileStringToChimeString` button and the Chimestring replaces the molfile string in the TextBox.

4. Click the `Render molfile` button and the molecule displays in the `Renderer` control with a highlighted collection. (Molfiles support the `HighlightColor` property for persistent collections, and we set this property to `Red`.)



5. Click the `Render SMILES file` button and the molecule displays in the `Renderer` control.



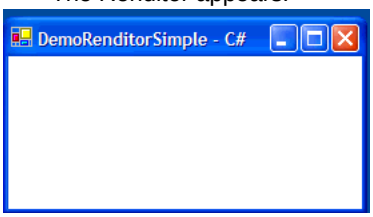
You have now built an application that uses the .NET components of Accelrys Draw.

DemoRenditorSimple Example

1. Start the Demo Renditor Simple example

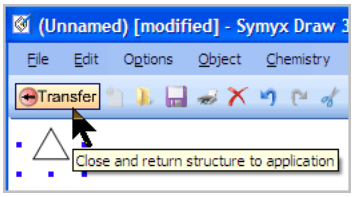
   `DemoRenditorSimple-csharp.exe`, is located in the `Examples\C#\DemoRenditorSimple\bin\Release\` subfolder of your Accelrys Draw installation. You can also launch it from the Start menu.

   The Renditor appears.

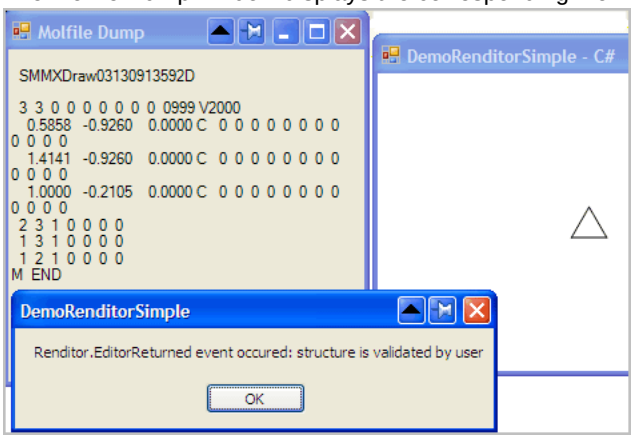


2. Double-click the canvas of the Renditor to launch the Editor.

3. Draw a structure in the Editor and click **Transfer**.



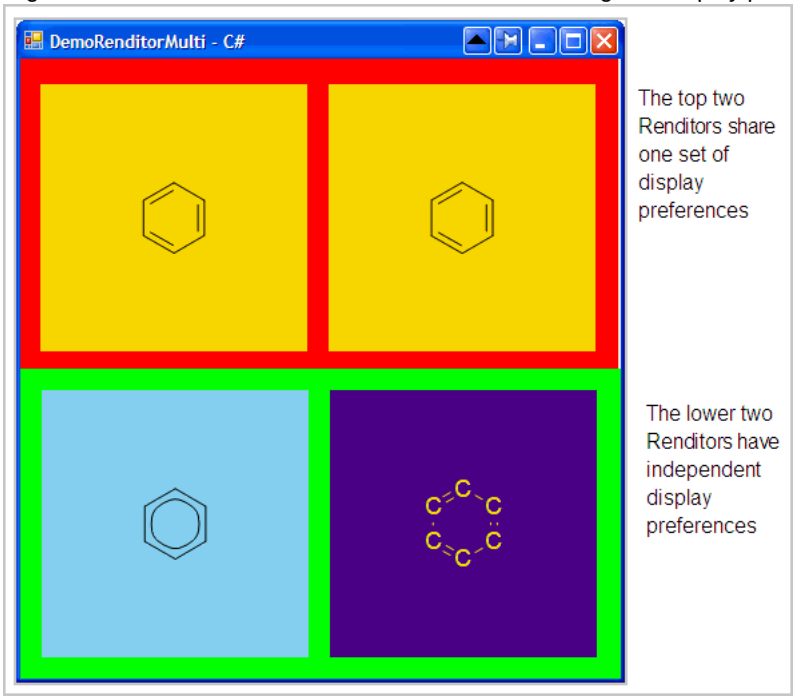
The Molfile Dump window displays the corresponding molfile.



Note: This example also indicates a valid structure or shows a Chemistry Check warning.

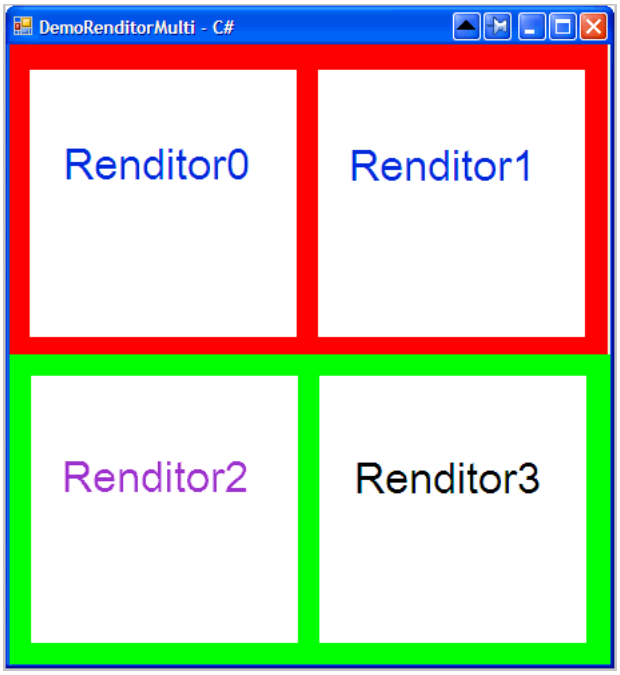
DemoRenditorMulti Example - a grid of renditors and sharing display preferences

A grid of four renditors, two of which use the same settings for display preferences.



The picture above shows that:

- `Renditor0` and `Renditor1` share the same settings for `BackColor` (yellow) and `AromaticRingsAsCircles` (false).
- `Renditor2` and `Renditor3` are independent of `Renditor0`, `Renditor1`, and each other.



To start the Demo Renditor Multi example

`DemoRenditorMulti-csharp.exe` is located in the `Examples\C#\DemoRenditorMulti\bin\Release\` subfolder of your Accelrys Draw installation. You can also launch it from the Start menu.

To synchronize Preference settings across renditors

To synchronize the display preference settings for multiple renditors, set their Preferences to the same object.

The Demo Renditor Multi example presents four Renditors in a grid. The two Renditors at the top (`renditor0` and `renditor1`) have synchronized DisplayPreferences because `DemoRenditor.cs` assigns both Renditors to the same Preferences object.

```
this.renditor1.Preferences = this.Renditor0.Preferences;
```

Any change to the display preferences of `Renditor1` also changes the display of `Renditor0` as soon as the `Transfer` button is clicked. Similarly, any change to the display preferences of `Renditor0` also changes the display of `Renditor1` as soon as the `Transfer` button is clicked.

Note: If two Renditors share one XML configuration file, at startup, both Renditors will look in the same location for the preferences file. However, during runtime, each Renditor's preferences could become different. Suppose you fire the Editor, then set the background color from white to yellow, and do not yet click `Transfer`. Until you click the `Transfer` button, the Renditors do not repaint, and the white background color remains. Clicking the `Transfer` button causes the Renditors are be repainted, so their new background color is set.

Special API Features

The Draw API for .NET include some special features, which are briefly described below. For details, see the Draw API Reference for .NET, a link for which is at API: Getting Started.

Determining if a structure is a reaction or a molecule

To determine whether a structure is a reaction, use the `IsReaction` property, which is on the Renderer and Renditor.

Note: Except for `IsReaction`, the Accelrys Draw API works with both reactions and molecules. For example, the `Render.MolfileString` property and the `StructureConverter.ChimeStringToMolfileString` method work with both reactions and molecules. Another example: the `Render.FileName` property works with rxnfiles the same way as it does with molfiles.

Structure editing: enabling and disabling

The `Renditor` has the `EditingEnabled` property, which specifies whether the current Renditor enables editing. If `true`, the end-user can launch an Editor by double-clicking and also right-click to get a context menu that allows edit and copy. If false, structures can only be rendered. Default: `true`.

Removing the rendered structure

You might want your application, at some point, to reset the rendering component to a blank canvas. Here is the syntax to render nothing.

```
Renderer.MolfileString = null;
```

Implicit carbons: setting highlight

To set the size of the dots that highlight atoms in a persistent collection, see `MDL.Draw.Renderer.DisplayPreferences.AtomHighlightDotWidth`.

Note: XML configuration can also perform this task. See the Accelrys Draw Configuration Guide on "Setting Values for Display Preferences".

Setting the XML configuration file for a Renditor

Your application can set a non-default XML configuration file for a Renditor. For API details, see `MDL.Draw.Renditor.Renditor.XMLConfig` in the *Draw API Reference for .NET*.

Example: How to set the primary XML config file using the API

In this example, we rename and edit the XML configuration file, and then we edit the `DemoRenditorSimple` code to use the custom XML configuration file.

1. Copy the `AccelrysDraw-net.xml` in the root directory your Draw installation.
2. Rename the copy `test.xml` and move `test.xml` to the root of your C drive.

3. Edit the XML configuration file to display in a different name in title bar of the Editor.

   ```
   <mdldraw width="950" height="600" name="Draw using test.xml as config file">
   ```

4. Open the solution file, `<draw_home>\Examples\C#\csharpexamples.sln`

5. Set `DemoRenditorSimple-csharp` as the start-up project.

6. Import the `System.IO` namespace (`using System.IO;`) and, in the DemoRenditorSimple constructor, add the following:

   ```
   // Specify the XML configuration file to use
   StreamReader sr = new StreamReader(@"c:\test.xml");
   renditor.XMLConfig = sr.BaseStream;
   ```

7. Build and run the project. When you double-click the Renditor, the Editor displays with the following text in the title bar: `Draw using test.xml as config file`

Note: See also Disabling Chem Check from Transfer back to Renditor, which gives an example of how to Specify the XML Configuration file the Renditor uses.

V3000 molfile output, query highlighting, and collections

By default, Accelrys Draw attempts to save molecules and reactions to V2000 format files to maximize interoperability with third-party applications. However, Accelrys Draw writes the molfile or rxnfile in V3000 format if there are any V3000 features or any features that exceed the fixed-field formats of the V2000 CTfile formats. For example,

- If the number of atoms (or bonds) exceeds 999, the V3000 format will be used.
- Query highlighting also involves collections and therefore forces the V3000 format.

For details, see the CTfile Formats document, which is available for download at http://accelrys.com/resource-center/downloads/.

- The Editor and Renditor have a display preference called `ForceV3000` to force the output to be in V3000 format. This is useful if you want all molfiles, even those without V3000 features, to be in one format. (It is not possible to store V3000 features in the V2000 format.) For a step-by-step example, see ForceV3000 API: step-by-step example. For the corresponding XML configuration, see To specify the value for a preference in the *Accelrys Draw Configuration Guide*.
- For a comprehensive discussion of Accelrys enhanced stereochemistry, see White Paper: Enhanced Stereochemical Representation at http://accelrys.com/products/informatics/cheminformatics/. Appendix A contains an example of a V3000 Molfile with enhanced stereochemistry for multiple stereocenters. This example involves collections, which forces the V3000 (rather than V2000) molfile format.

ForceV3000 API: step-by-step example

These steps show using the API to force all molfiles to be written in V3000 format.

1. Open the `<DRAW_HOME>\Examples\C#\csharpexamples.sln`, where `<DRAW_HOME>` represents the root directory of your Accelrys Draw installation.

2. Set `DemoRenditorSimple-csharp` as the `StartUp` project.

3. In the code of `DemoRenditorSimple.cs`, add a final line to the constructor that sets the `ForceV3000` preference to `true`.

```
public RenditorDemoSimple() {
   InitializeComponent();
   this.renditor.Preferences.ForceV3000 = true;
   // ...
        }
```

4. Rebuild the project.
5. Start the project.
6. Double-click the Renditor to launch the Editor.
7. Add an atom and save the structure as a molfile.
8. Open the molfile in a text editor and you will see the V3000 indicator at the end of line 4. For example:

```
  SMMXDraw06200815532D

  0  0  0     0  0            999 V3000
```

instead of the V2000 indicator:

```
  SMMXDraw06200815202D

  5  5  0  0  0  0  0  0  0  0999 V2000
```

Disabling Chem Check from Transfer back to Renditor

By default, when the user clicks the `Transfer` button, Accelrys Draw Renditor invokes a Chem Check. If you want to disable this feature, perform these two tasks:

- Edit the XML Configuration file for donetoolbar
- Specify the XML Configuration file the Renditor uses

Edit the XML Configuration file for donetoolbar

1. Open `<draw_home>\XmlConfig]donetoolbar.xml`
2. Make a copy of this file and rename the copy to `MyDoneToolbar.xml`
3. Edit `MyDoneToolbar.xml`

The XML looks like this BEFORE you make the edit:

```
<donetoolbar>
   <action image="Done.gif" class="MDL.Draw.Modules.Editor.Actions.DoneAction"
      tooltip="Transfer to the Web">
      <action name="QueryChemInspector"
class="MDL.Draw.Modules.Editor.Actions.Calcs.QueryChemInspectorAction">
      <dialog title="Chemistry Check Warning" command="querycheck">
            <space height="7" />
            <panel>
               <label text="#" command="result" update="T" variable="T" />&apos;
            </panel>
            <space height="7" />
            <panel>
               <label text="Use Chemistry Check" command="use" />&apos;
               <label text="to step through problems." command="problem" />&apos;
            </panel>
            <space height="7" />
            <panel layout="horizontal" align="center" width="250" height="40">
               <button text="&amp;OK" command="ok" />
               <button text="Continue &amp;Transfer" command="continue" width="115"
/>
            </panel>
         </dialog>
      </action>
   </action>
</donetoolbar>
```

The XML looks like this AFTER you make the edit:

```
<donetoolbar>
   <action image="Done.gif" class="MDL.Draw.Modules.Editor.Actions.DoneAction"
           tooltip="Transfer to the Web">
   </action>
</donetoolbar>
```

4. Save `MyDoneToolbar.xml`.
5. Navigate up one directory.

6. Make a copy of `AccelrysDraw-net.xml`
7. Rename this copy to `MyAccelrysDraw-net.xml`
8. Edit `MyAccelrysDraw-net.xml` so that the donetoolbar line reads as follows:

```
<donetoolbar include="MyDoneToolbar.xml"/>
```

9. Save and close `MyAccelrysDraw-net.xml`.

Specify the XML Configuration file the Renditor uses

To make the Renditor use a custom XML Configuration file, specify the XML Configuration file (including its path from the executable) for the constructor to read in from the file system.

For example, add the following lines of code:

```
            System.IO.FileStream fs = System.IO.File.OpenRead(@"C:\Program Files
(x86)\Accelrys\Accelrys Draw 4.1\MyAccelrysDraw-net.xml");
            renditor.XMLConfig = fs;
```

where `MyAccelrysDraw-net.xml` is the name of the primary XML Configuration file that points to `MyDoneToolbar.xml`.

What follows is an example constructor in context.

```
namespace DemoRenditorSimple {
    /// <summary>
    /// RenditorDemo description
    /// </summary>
    public class RenditorDemoSimple : System.Windows.Forms.Form {
        private MDL.Draw.Renditor.Renditor renditor;
        private IContainer components;
        MolfileDisplayForm molfileDisplayForm  = new MolfileDisplayForm();


        public RenditorDemoSimple() {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();
            System.IO.FileStream fs = System.IO.File.OpenRead(@"C:\Program Files
(x86)\Accelrys\Accelrys Draw 4.1\MyAccelrysDraw-net.xml");
            renditor.XMLConfig = fs;
        }
[...]
```

Java Native Interface - JNI Wrapper

The Java Native (JNI) Wrapper example allows a Java application to have access to the Accelrys Draw Renderer and Renditor. The example is located at `<draw_home>\Examples\Java`.

The API Reference for the wrapper class, `com.mdli.draw.JMolRendererCom`, is located at `<draw_home>\docs\Draw\api-java\index.html`

You can use the Swing JPanel version or the AWT version. In either case, follow these instructions.

1. Make sure your computer has an installation of BOTH of the following, which are available at `http://java.sun.com`:
   - Java Runtime Environment (JRE) version 1.4 or later
   - a compatible Java Development Kit (JDK)
2. Install Accelrys Draw for .NET
3. Make sure your PATH environment variable references the `<draw_home>\lib` folder for Draw.
4. Make sure your PATH environment variable references *your version number* of the Java *Runtime* Environment (JRE) so you can run the example.

```
C:\Program Files\Java\jre1.4_2_16\bin;
```

where `jre1.4_2_16` is one example that includes a version number of the JRE.

5. Make sure your PATH environment variable references, as well, *your version number* of the Java *Development* Kit (JDK) so that you can compile.

```
C:\Program Files\Java\jdk1.4_2_16\bin;
```

where `jdk1.4_2_16` is one example that includes a version number of the JDK.

6. Append to your CLASSPATH environment variable the path to `JMolRendererCom.jar`. For example,

```
<draw_home>\lib\JMolRendererCom.jar;
```

7. Append to your CLASSPATH environment variable the path to the current directory. To this add the period or dot (`.`). For example,

```
.;
```

8. Reboot the machine.

Swing JPanel version

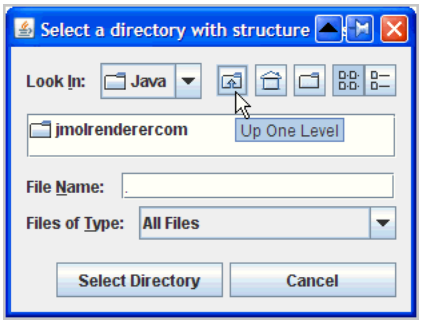
This version places the AWT code onto a Swing JPanel.

1. Compile from above the package, which by default means:

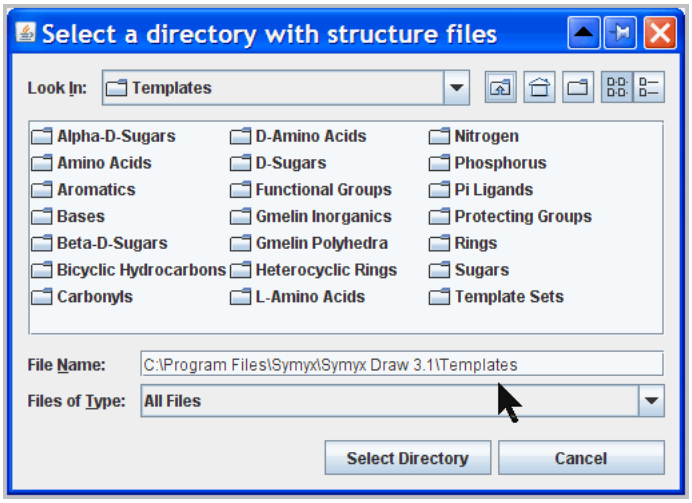
```
<draw_home>\Examples\Java>javac JMolRenderercom/Swing/*.java
```

2. Run the sample application:

```
<draw_home>\Examples\Java>java jmolrenderercom.Swing.TestPanel
```

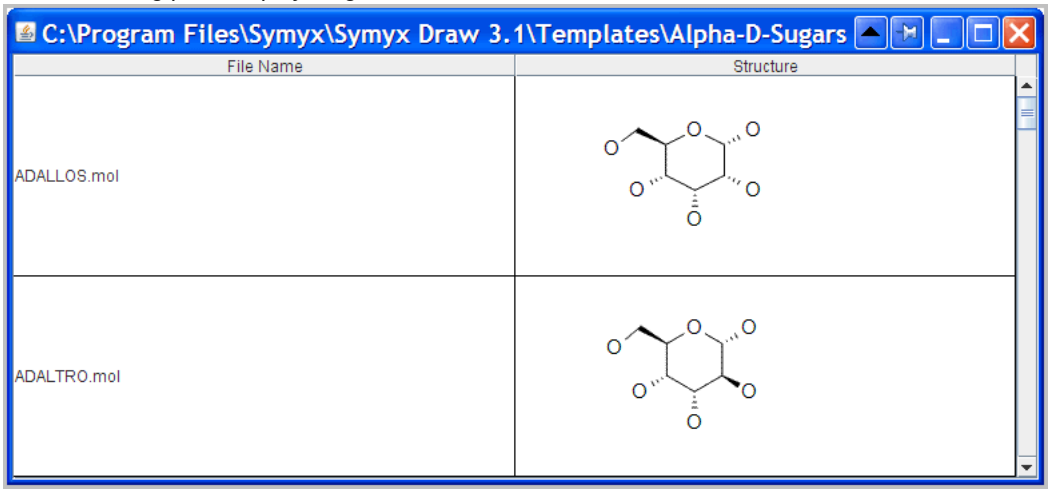


3. Navigate to a folder that has structure files.



4. Click Select Directory.

   The Swing pane displays a grid.



Note: You can add code to the cells to that double-click will launch the Editor.

AWT version

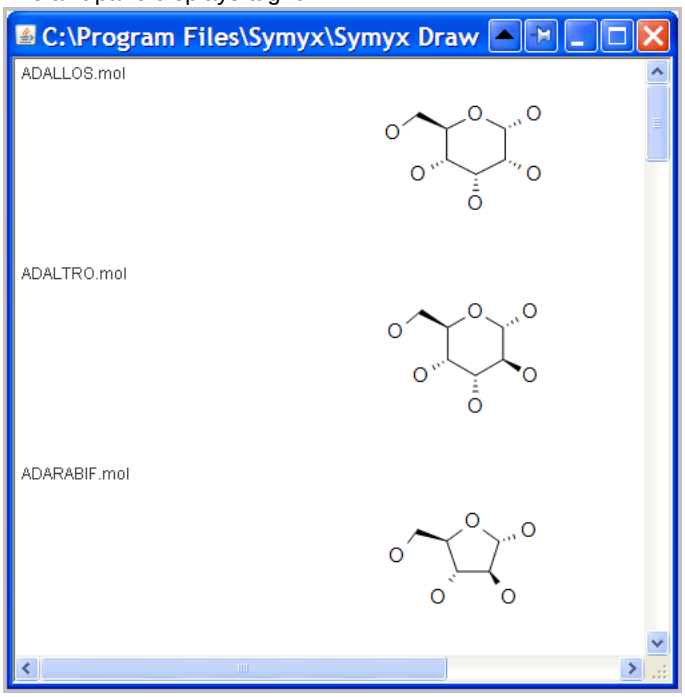
1. Compile from above the package, which by default means:

```
<draw_home>\Examples\Java>javac JMolRenderercom\awt\*.java
```

2. Run the sample application:

```
<draw_home>\Examples\Java>java jmolrenderercom.awt.TestRendererScrollPane
```

The awt pane displays a grid.



Note: You can add code to the cells to that double-click will launch the Editor.

Custom Ptable for a JMolRendererCom Java application

The Accelrys Draw Renditor, when called from the .NET or JNI interface, can load the settings defined in the `AccelrysDraw-net.xml` configuration file.

To maintain backward compatibility with existing applications, Accelrys has implemented this feature as a setting that can be configured in the Windows Registry as well as through a new API method exposed in the Java interface. See JMolRendererCom API Method

When the Renditor is loaded, it looks for the registry property `RenditorUseLocalXMLConfig` on the following key:

`HKEY_CURRENT_USER\Software\Symyx Technologies, Inc.\Symyx Draw\Client\`*`[version number, such as 4.1]`*

If `RenditorUseLocalXMLConfig` is not found in this location, the Renditor looks for it at:

`HKEY_LOCAL_MACHINE\SOFTWARE\Symyx Technologies, Inc.\Symyx Draw\Client\`*`[version number]`*

If the key property is found in either location, the Renditor compares its value with the following values:

Key Value	Behavior to initialize the editor
Always	Always use `<Draw installation directory>\AccelrysDraw-net.xml>`
Implicit	Use `<Draw installation directory>\AccelrysDraw-net.xml>` unless an XML stream is set explicitly through the API
Not set or other value found	Predefined within the Renderer and Renditor DLLs unless the `Renditor.UseLocalXMLConfig` boolean field is set to `true`. This allows existing applications to continue to function as they now do.

Note:

- For the Java interface, the behavior described above can be overwritten using the API method described as JMolRendererCom API Method.
- If `Renditor.UseLocalXMLConfig` is set to `true`, Accelrys Draw grants this request even if an xml stream was provided or the `RenditorUseLocalXMLConfig` registry key is set.

JMolRendererCom API Method

The `JMolRenderCom` API allows the setting in `AccelrysDraw-net.xml` to be more easily loaded when initializing the Renditor.

`The Renditor.UseLocalXMLConfig` property accepts a Boolean value. When set to `true`, this function loads the settings defined in `AccelrysDraw-net.xml`. Make this initialization immediately after the Renditor is created and before it is displayed.

```
JMolRendererCom renderer = new JMolRendererCom();
renderer.SetUseLocalXMLConfig(true);
```

Note: If `Renditor.UseLocalXMLConfig` is set to `true`, Accelrys Draw grants this request even if an XML stream was provided or the `RenditorUseLocalXMLConfig` registry key is set.

Facade API

The Facade API allows a customer application to programmatically manipulate molecules and sketches of molecules.

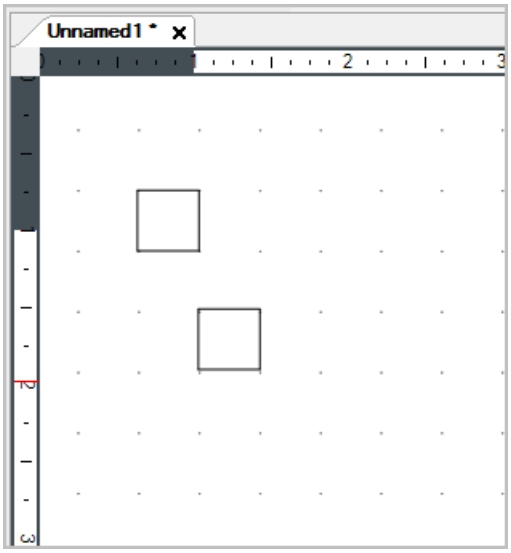
The namespace in the API Reference is `MDL.Draw.Editor.Facade`.

An example is installed at `<draw_home>\Examples\C#\FacadeAPIActionExample`.

Note: The Facade API does not support the manipulation of reactions, Rgroups, or Sgroups.

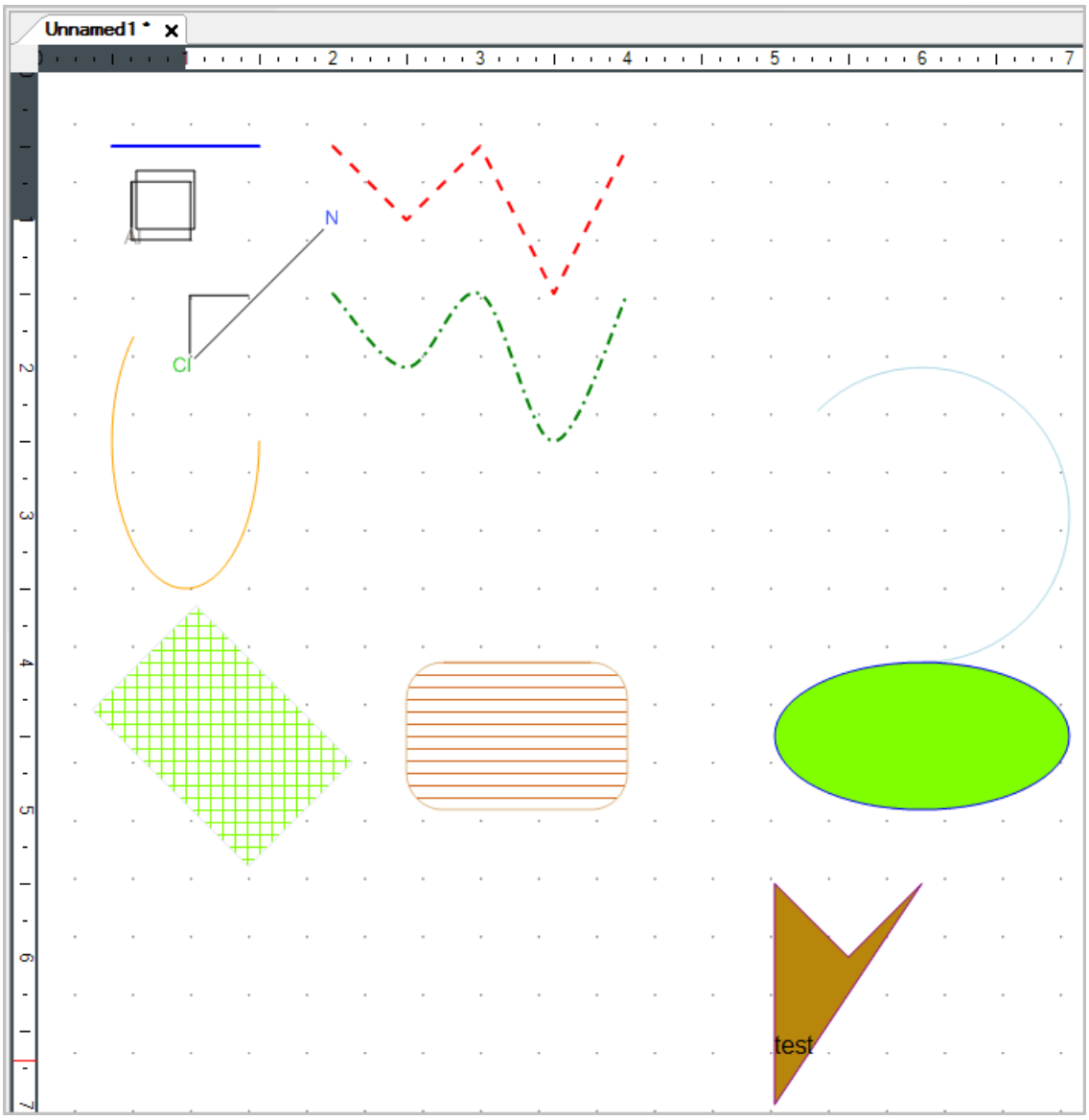
To run the example,

1. To add an action to the Chemistry menu, copy `<draw_home>\Examples\C#\FacadeAPIActionExample\FacadeApiAction.xml` into `<draw_home>AddIns\`
2. Restart Draw.
3. Add two cyclobutanes to the canvas.



4. On the `Chemistry` menu, click `Facade API Demo`.

5. Note that various programmatic modifications and additions have occurred, both chemically-significant and sketch (such as the ellipse). The code for these manipulations is in `FacadeApiAction.cs`

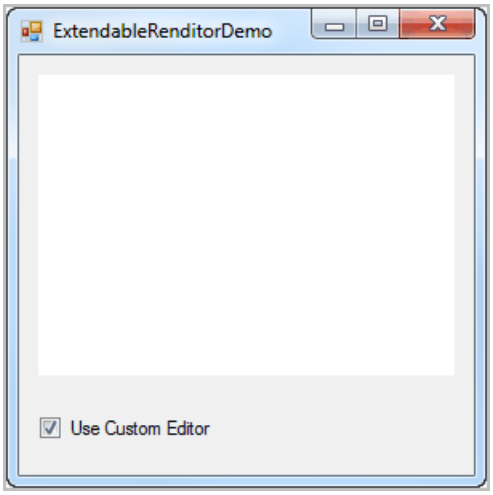


ExtendableRenditorDemo

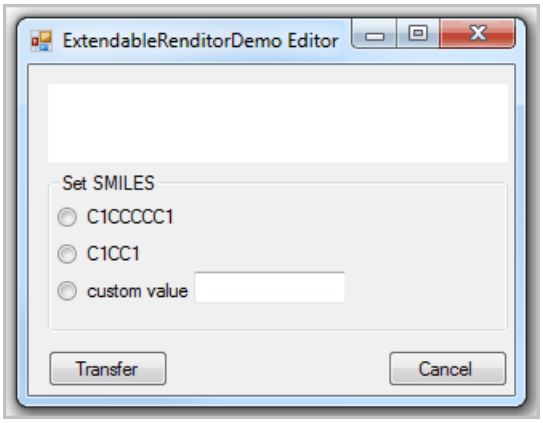
You can customize the Renditor such that it launches a custom Editor instead of the standard Draw Editor. The code for this example is installed at `<draw_home>\Examples\C#/DemoExtendableRenditor`. In particular, `ExtendableRenditorDemoForm.cs` shows how the Renditor can be set to use a customized Editor:

```
extendableRenditor1.EditExtension = new MyEditor();
```

1. From the Start menu, click `All Programs > Accelrys > Accelrys Draw 4.1 > Examples > C# Examples > DemoExtendableRenditor`.



2. To use the standard Draw Editor instead of the customer Editor, uncheck `Use Custom Editor`. Note that the default is to use the custom Editor.
3. Double-click the canvas.



4. Set a SMILES string and see it rendered on the canvas of the Editor.
5. To return to the Renditor with the structure, click `Transfer`.

StructureResolver API

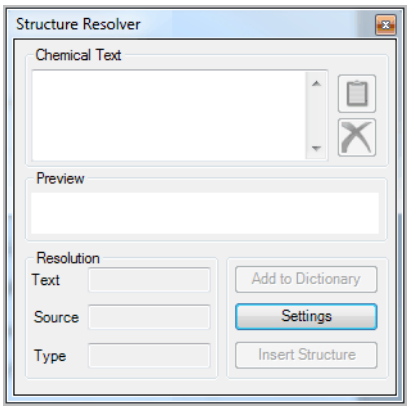
The Facade API allows a customer application to programmatically manipulate molecules and sketches of molecules.

The namespace in the API Reference is `MDL.Draw.Editor.EditorControl.StructureResolver`.

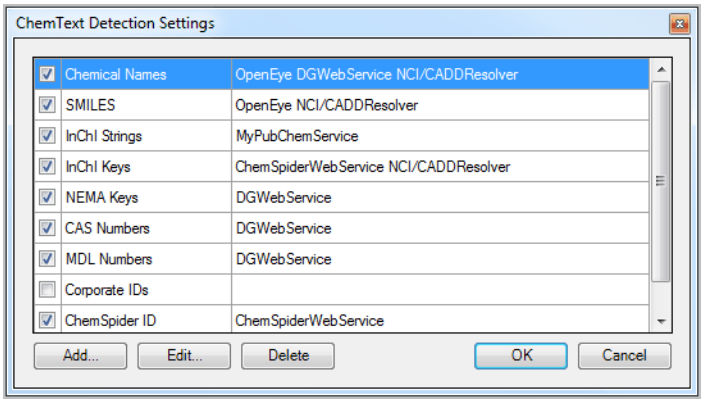
An example is installed at `<draw_home>\Examples\C#\DemoStructureResolver`

Deployment of the example

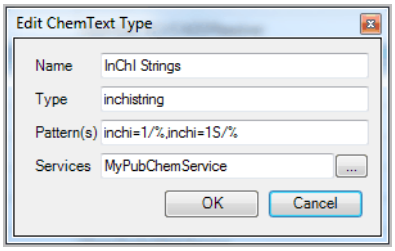
1. Copy `DemoStructureResolver.dll` from `<draw_home>\Examples\C#\DemoStructureResolver\obj\Release` to `<draw_home>\lib\StructureResolverServices`
2. Start Accelrys Draw.
3. On the **`Chemistry`** menu, click Structure Resolver. The **`StructureResolver`** dialog displays.



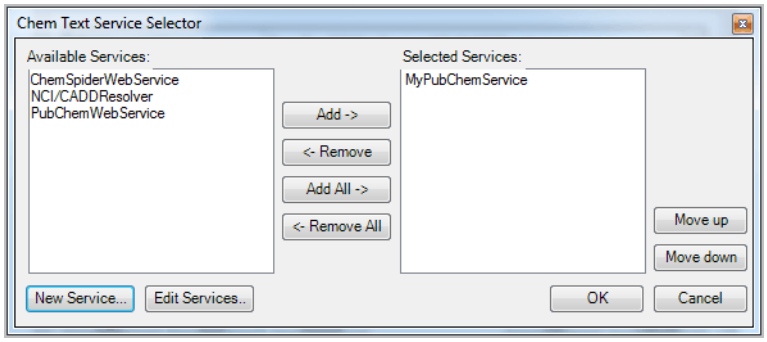
4. Click `Settings`, then `Resolver Settings`. The `ChemText Detection Settings` dialog displays.



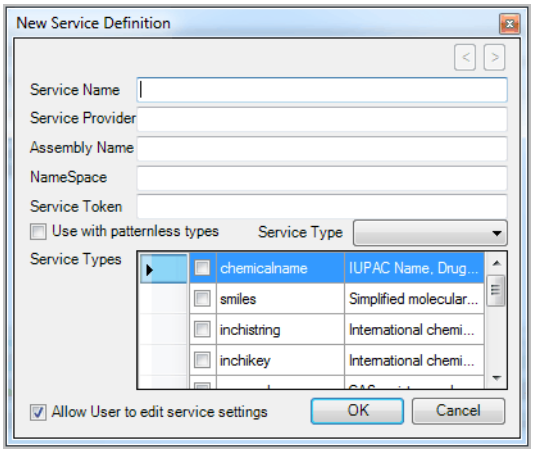
5. Click `InChI Strings`, then `Edit`. The `Edit ChemText Type` dialog displays.



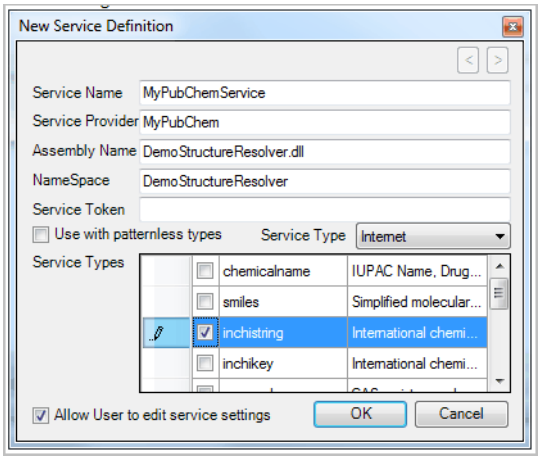
6. Click the button with "`. . .`" that is near the `Services` label. The `Chem Text Service Selector` dialog displays.



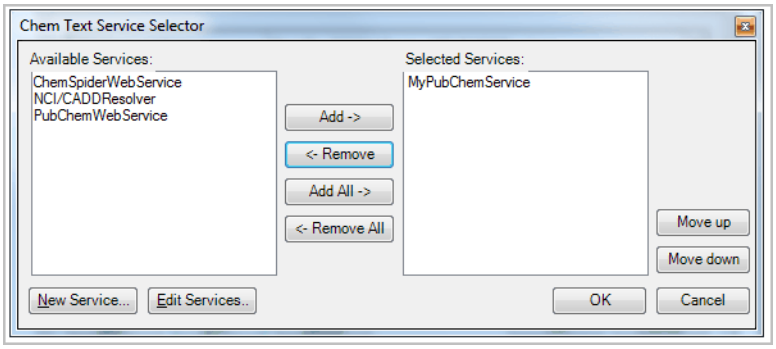
7. Click `New Service.` The `New Service Definition` dialog displays.



8. Set `Service Name` to MyPubChemService, set `Service Provider` to MyPubChem, set `Assembly name` to DemoStructureResolver.dll, and set `NameSpace` to DemoStructureResolver. Also, make sure that `Service Types` shows `inchistring` as checked.

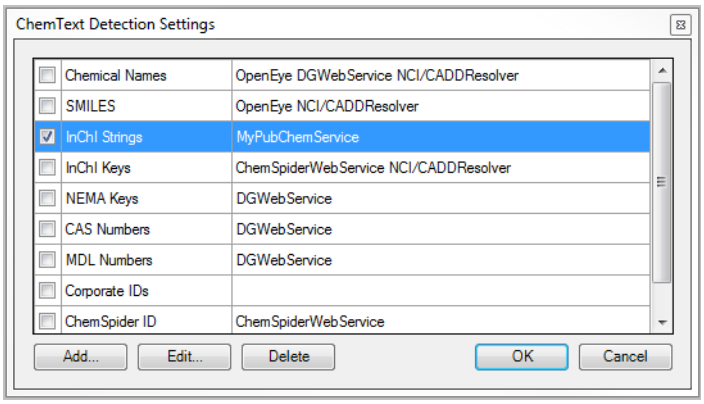


9. Click OK. The `Chem Text Service Selector` dialog displays.

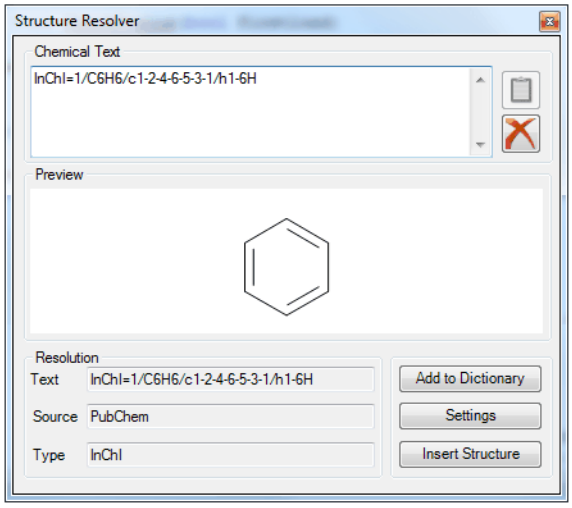


10. Add MyPubChemService to `Selected Services` and remove any other services.
11. Click OK. The `Edit ChemText Type` dialog displays.
12. Click OK. The `ChemText Detection Settings` dialog displays.

13. Make sure that `InChi Strings` is checked and no others are checked.

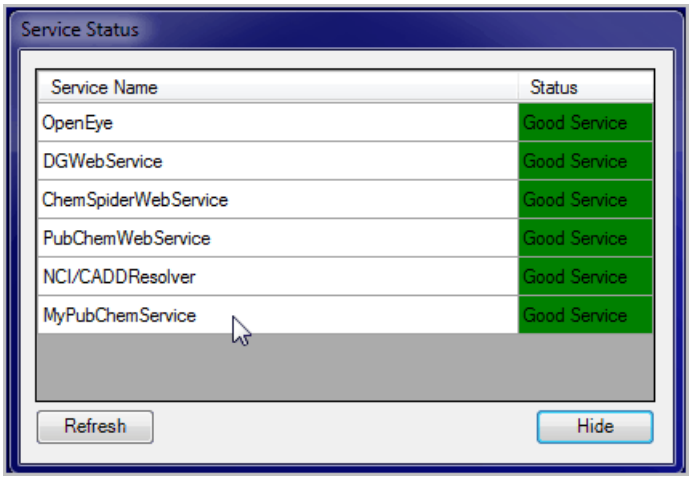


14. In the Structure Resolver dialog, under Chemical Text, enter an InChI string to test, such as `InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H` for benzene.



15. Verify that corresponding structure is displayed.

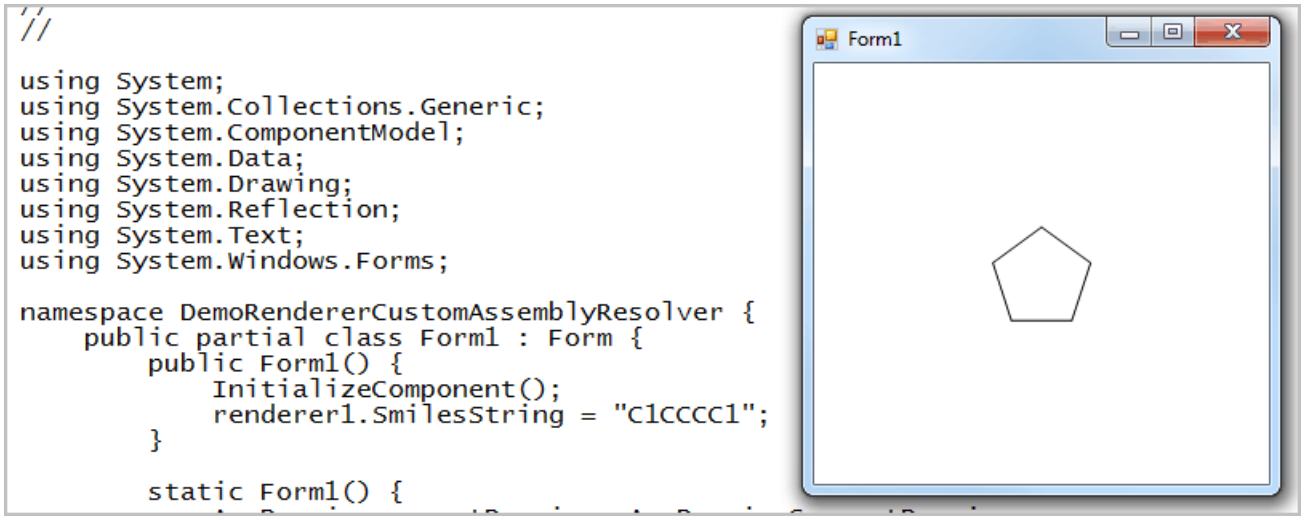
16. Note that you can verify that a service is working. In the Structure Resolver dialog, click Settings > Service Status.



DemoRendererCustomAssemblyResolver

To support one code stream supporting both Draw ClickOnce and Draw MSI, use the example located at `<draw_home>\Examples/C#/DemoRendererCustomAssemblyResolver/`

Whereas ClickOnce creates a user environment variable, `DRAW_DEPLOY_PATH`, that points directly to the ClickOnce location of the Draw executable and libraries, the 4.1 MSI installer places the Draw libraries in the `lib` subdirectory of the location referenced by this user environment variable. Therefore, an application that considers both `DRAW_DEPLOY_PATH` and `DRAW_DEPLOY_PATH\lib` can target the Draw 4.1 executable whether Draw was installed by ClickOnce or MSI.



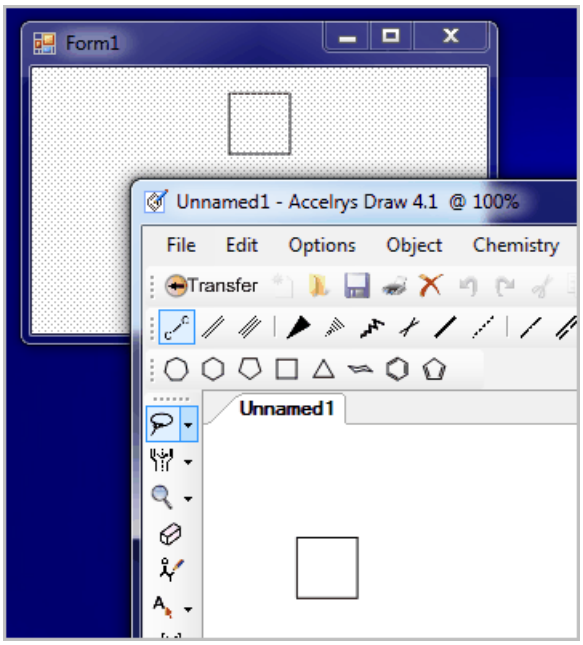
In particular, Form1.cs contains the following code to handle both situations:

```
try {
    // if not found yet, try to load from DRAW DEPLOY PATH
    // which is the ClickOnce location.
    assembly = Assembly.LoadFile(path + "\\" + requestedassemblyname);
}
catch (Exception) {
    if (assembly == null) {
try {
    // if not found yet, try to load from DRAW DEPLOY PATH\lib
    // which is the MSI location.
    assembly = Assembly.LoadFile(path + "\\lib\\" + requestedassemblyname);
}
```

DemoLateBoundRenditor

Your application might not know at compile time whether the end-user has the click-once or the MSI version of Draw. Therefore, you might need to be prepared for either situation at runtime. To support launching the click-once version of Draw if the MSI version is not found, look at the code in `<draw_home>\Examples\C#\DemoLateBoundRenditor`, in particular, `RenditorForm.cs` which has the following:

```
try {
    // The MSI version uses the lib subdirectory
    editorassembly = Assembly.LoadFrom(deploypath + @"\lib\MDL.Draw.Editor.dll");
}
catch {}
if (editorassembly == null) {
    try {
// If no MSI, try to use click-once version of Draw
editorassembly = Assembly.LoadFrom(deploypath + @"\MDL.Draw.Editor.dll");
    }
```



Add-ins: Actions and Tools

About Add-in Actions

An add-in (also known as an addin) enables the extension of Accelrys Draw functionality with software from other sources.

Add-in actions are associated with a menu item. It is also possible to have an Add-in associated with a tool.

You can also add your own Add-ins at Add-in Action Example: DisplaySmilesStringAction.

Note:

- Draw API Reference for .NET contains interfaces to use with Add-ins. See the `MDL.Draw.Interfaces` namespace.
- To hide an add-in, see the Accelrys Draw Configuration Guide on “Example: Removing from the Chemistry menu”.

Add-in Action Example: DisplaySmilesStringAction

This walkthrough provides `DisplaySmilesStringAction` as an example of how to integrate Add-in actions into Accelrys Draw. The example is included here as an illustration of tasks that you, the Accelrys Draw application developer, are likely to perform for your own Add-in actions.

Note:

- For Accelrys Draw API support of SMILES, see `Draw API Reference for .NET`, especially `MDL.Draw.HeadlessRenderer.StructureType.Smiles` and `MDL.Draw.StructureConversion`.
- This example is only a demonstration and is distinct from the Accelrys Draw built-in capability to generate SMILES, which is one of the options already available from the Chemistry menu > Generate Text from Structure.

Files for the Add-in action

To integrate Add-in actions, you need the following files:

- An XML configuration file for your Add-in.

  When Accelrys Draw starts, it configures Add-ins by reading the .xml files in the `<draw_home>\AddIns\` directory. This XML Configuration file:

  - Defines the menus and menu items that run the Add-ins
  - Specifies the path from `AccelrysDraw-net.exe`, that is, the `<draw_home>` folder, to the .NET assembly and class that implement the Add-in action.
  - Specifies the name of the .NET assembly and class that implement the Add-in action.
  - Can specify more than one Add-in.

  - In this example, `DisplaySmilesString.xml` is the name of the XML configuration file for the Add-in. For details, see Editing the DisplaySmilesString.xml file.

- The .NET assembly that implements the Add-in.

  This .NET assembly must define a class that implements the Draw `IAction` interface. When Draw starts up, Draw instantiates the Add-in action class. When the end-user selects the menu item associated with the Add-in, Draw invokes the `ActionPerformed` method of the Add-in action class.

Setting up the AddIns directory

At startup, Accelrys Draw automatically checks for any add-ins in the `AddIns` subdirectory of your Accelrys Draw installation.

1. Verify that a directory named `AddIns` is a subdirectory of your Accelrys Draw installation. The default location is `<draw_home>\AddIns\`
2. Navigate to the `<draw_home>\Examples\C#\AddInActionExample` subdirectory of the root directory of your Accelrys Draw installation
3. Copy `DisplaySmilesString.xml` into the `[draw-home]\AddIns` subdirectory

Note: The DLL that Accelrys Draw calls to access the Add-in is **`MDL.Draw.AddIn.DisplaySmilesString.dll`**, which is located in the **`Examples\C#\AddInActionExample\bin\Release`** directory of your installation.

Editing the DisplaySmilesString.xml file

The `DisplaySmilesString.xml` file has the following elements and attributes:

`mdldraw` - Top-level element with two attributes.

`debug` - (optional) If `"true"`, generates error file for every unhandled Accelrys Draw error (not only Add-in action errors), and writes this text to a file that increments its number with each appearance: `MDLDrawBugReport0.txt`, `MDLDrawBugReport1.txt`, `MDLDrawBugReport2.txt` and so on. Error files are written to the Accelrys Draw installation directory.

`popupsErrors` - (optional) If `"true"`, displays each error on the screen as a message box instead of writing a file.

`menu` - Specifies an existing menu or new menu as the location for the add-in. The `menu` element has one attribute:

`name` - (required) The name of an existing menu

To specify a menu and submenu, repeat the element. For example:

```
<menu name="Special">
  <menu name="Add-ins">
```

`action` - Defines the menu line: the display text, the DLL to call, the class in that DLL that implements the `IAction` interface, and optionally the position of this line in the menu. It has the following attributes:

`name` - (required) The text to display for the menu item

`addInAssemblyPath` - (required) The path and assembly that contains the class that implements the `IAction` interface. The path can be relative to either <draw_home> or <draw_home>`\AddIns`

`class` - (required) The class in the specified assembly that Accelrys Draw will instantiate at start up. This class implements the `IAction` interface and defines the `ActionPerformed` method. (Accelrys Draw invokes the `ActionPerformed` method when the user selects this menu item.)

`index` - (optional) The position for this menu line in an existing menu. If `index` is not specified, the position is at the top, as the first menu item. To override this default, specify an `index`, such as 2.

`image` - (optional) The image to display with the tool or menu item. Make the image an embedded resource of the project. For an example, see About Add-in Tools.

Note: An Add-in can have one or more `action` elements.

Specifying the path

The relative path is from the root directory of the Accelrys Draw installation or from the AddIns subdirectory:

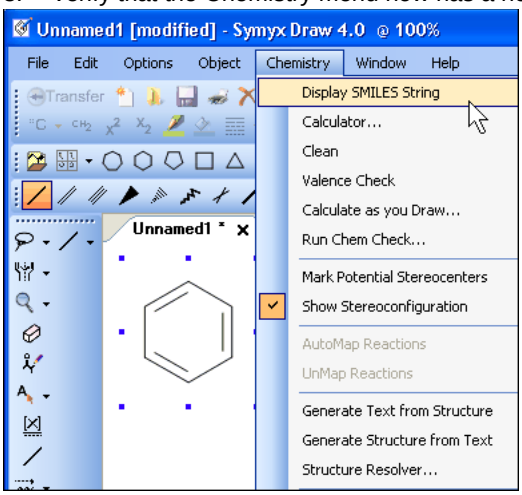
```
<mdldraw>
<!--This file should be copied to the AddIns directory.-->
<!--Add this action to the Chemistry menu-->
<menu name="Chemistry">
<!--Specify display string and position, class, and assembly-->
<action name="Display SMILES String"
index="1"
class="MDL.Draw.AddIn.DisplaySmilesStringAction"
addInAssemblyPath="Examples/C#/AddInActionExample/bin/Release/MDL.Draw.AddIn.Display
SmilesString.dll"/>
</menu>
</mdldraw>
```

Testing the example

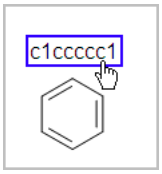
Test the example by using the `DisplaySmilesString.xml` file shown above.

1. Restart Accelrys Draw.
2. Add a molecule to the canvas.

3. Verify that the Chemistry menu now has a new item, `Display SMILES String`.



4. Click the Display SMILES String menu item. The corresponding SMILES string displays.



Customizing the menu

In this example, `index="1"` means that the new menu item is at the top of the menu. To set a different location, change the value of the index. For example, index="3".

One way to add the menu line to the bottom of an existing menu is to omit the `index` attribute of the `action` element.

Removing an Add-in Action

To remove an Add-in Action, edit your XML configuration file for the Add-in to either remove or comment out the entry, and then restart Accelrys Draw.

Implementing the IAction Interface for Your Add-in Action

The following are guidelines to follow when you create your own Accelrys Draw Add-in action by implementing the `IAction` interface. For an example, see `Addin.cs` in your `Examples` directory.

1. Declare that your Add-in action class implements `IAction`.

   ```
   public class MyAddinAction : IAction
   ```

2. Create a class that implements the `IAction` interface.

3. Use the line ending termination format appropriate for the operating system: `Renderer.MolfileStringDOS`, `Renderer.MolfileStringMacOS`, `Renderer.MolfileStringUNIX`.

4. If your add-in action displays text along with the rendered structure, call `AddTextToStructure`.

   ```
   editor.AddTextToStructure(variableForTextToDisplay);
   ```

5. Read the API Reference documentation for `MDL.Draw.Interfaces.IEditor`, which provides Add-ins with a handle to the Accelrys Draw Editor.

About Add-in Tools

It is likely that most Add-ins will be associated with an action, such as the user clicking on a menu item (see About Add-in Actions). Therefore, this section assumes you have read About Add-in Actions because much of its information applies to tools, for example, the elements of the XML file. However, it is also possible to associate an Add-in with a tool, which this example will demonstrate.

Note: Accelrys Draw API Reference for .NET contains interfaces to use with Add-ins. See `MDL.Draw.Interfaces`.

Add-in Tool Example: Identify atom, bond, or Sgroup the end-user selected

This example adds a tool that does the following:

- Displays the V3000 molfile of the current structure
- Includes in the molfile a collection that indicates which atom, bond, or Sgroup the end-user selected

One scenario for such a tool might be an application that runs an Accelrys Cheshire script if the end-user selects an atom or bond that has attached data (data Sgroup).

Accelrys Draw includes an example you can see from the VB.Net examples solution. Assuming that you are in the root directory of your Accelrys Draw installation, the relative path is `Examples\VB.Net\AddInToolExample-VB.Net\AddInTool.vb`

Note: You can also write the equivalent code using C#.

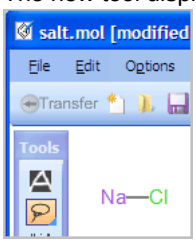
To run the example out-of-the-box

1. Create a file called `tool.xml` in the `<draw_home>\AddIns` folder with the following contents.

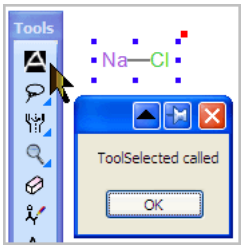
```
<mdldraw debug="true">
  <tool name="Molfile Popup Tool"
        class="AddInToolExample_VB.Net.DemoTool"
        image="AddInToolExample_VB.Net.addin20-20.gif"
        index="1"

addInAssemblyPath="Examples\VB.Net\AddInToolExample-VB.Net\bin\AddInToolExample-VB.N
et.dll"/>
</mdldraw>
```

2. Start Accelrys Draw. In this case, we also open a file named `salt.mol`.

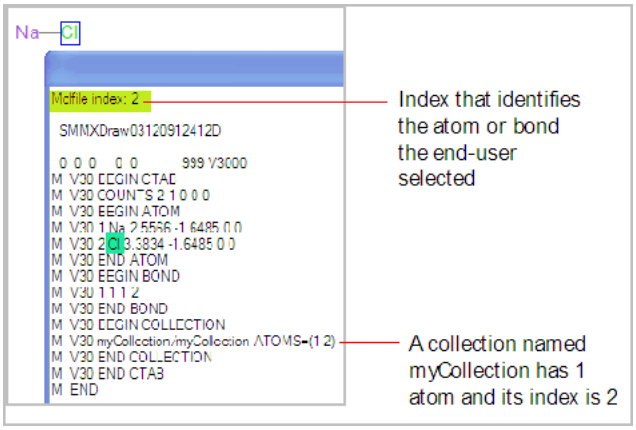
The new tool displays in at the index="1" location, that is, at the top.



3. Click the new tool. A message box displays.



4. Click an atom. In this case, we click chlorine, which is uniquely identified in the molfile as 2.



To conclude, this tool numerically identifies the atom or bond that the user selected and provides the molfile.

Note:

- The molfile contains a section called `COLLECTION` because the molfile contains a persistent collection that gives the index to the chemical object that the end-user selected. The name of the collection was assigned by this call: `chemObject.GetMolfileStringWithCollection("myCollection")`
- This add-in can also give the index of the Sgroup that the end-user selects.
- For details about molfiles, download the `CTFile Formats` document from `http://accelrys.com/products/informatics/cheminformatics/ctfile-formats/no-fee.php`

- For an example that uses the API to allow end-users to undo an action, see Undo incrementally for action or tool.

Making your Add-in Tool Project

Here are the steps for making and testing your own Add-in tool project.

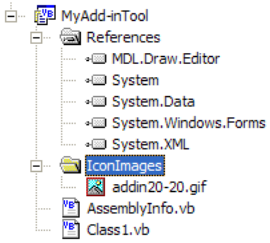
1. Create a new VB.NET project as a Class Library. Name the project `MyAdd-inTool`.

   If you create this project from the VB.NET examples solution, the project could be `\Examples\VB.Net\MyAdd-inTool\MyAdd-inTool.vbproj`.

   The VB.NET root namespace is `MyAdd_inTool` with an underscore (_) instead of `MyAdd-inTool` with a hyphen (-). That detail is necessary to configure the `tools.xml` file correctly in Step 8.

2. Add a reference to this .NET assembly: `System.Windows.Forms`
3. Add a reference to this .NET assembly: `System.Windows.Drawing`
4. Add a reference to the `MDL.Draw.Editor.dll`, which is located in the `lib` subdirectory of your Accelrys Draw installation.
5. Add a reference to the `MDL.Draw.Foundation.dll`, which is located in the `lib` subdirectory of your Accelrys Draw installation.
6. Add an icon for your tool as an embedded resource. For details, see Microsoft's documentation.

   In this example, we create a folder called `IconImages`, and add an item (a 24 x 24 pixel gif), and set the **`Build Action`** property of the image file to `Embedded Resource`.



7. Insert this implementation of the `ITool` interface into your project's `.vb` file:

```
Imports System.Windows.Forms
Imports System.Diagnostics
Imports System.IO
Imports System
Imports System.Drawing

Imports MDL.Draw.Interfaces
Imports MDL.Draw.Utils

    Class DemoTool
        Implements ITool

        Private myEditor As IEditor
        Public Overridable Sub Init(ByVal editor As IEditor) Implements
ITool.Init
            myEditor = editor
        End Sub

        Public Overridable Function ToolInvoked(ByVal chemObject As IChemObject,
ByVal e As MouseEventArgs) _
           As Boolean Implements ITool.ToolInvoked

            If Not (chemObject Is Nothing) Then
                Console.WriteLine(chemObject.MolfileIndex)
                Dim D As Dialog1
                D = New Dialog1

                D.TextBox1.Text = "Molfile index: " & chemObject.MolfileIndex & _
            Environment.NewLine &
MDL.Draw.Utils.DrawUtil.Unix2Dos(chemObject.GetMolfileStringWithCollection("myCol
lection/myCollection"))
                D.ShowDialog()

                ' If I were using a Form, I would have to set its Owner property
                ' m_form.Owner = myEditor.ApplicationForm
                Return True
            Else
                Return False
            End If
        End Function

        Private firstTime As Boolean = True
        Public Overridable Sub ToolSelected() Implements ITool.ToolSelected
            If (firstTime) Then
                MessageBox.Show("ToolSelected called")
            End If
        End Sub
```

```
        Public Overridable Sub ToolUnselected() Implements ITool.ToolUnselected
            If (firstTime) Then
                MessageBox.Show("ToolUnselected called")
                firstTime = False
            End If
        End Sub

        Dim myObjectArray As [Type]() = {GetType(IAtom), GetType(IBond),
GetType(ISgroup)}

        ' Signal which ChemObject types we are interested in
        ' The Draw application will highlight these, but not others.
        Public Overridable ReadOnly Property Targets() As Type() Implements
ITool.Targets
            Get
                Return myObjectArray
            End Get
        End Property

    End Class
```

8. Build your project and verify that you now have an assembly called `MyAdd-inTool.dll` in `Examples\VB.Net\MyAdd-inTool\bin`.
9. Copy `MyAdd-inTool.dll` to `<draw_home>\AddIns`.

   Note: Moving the DLL is not required, but in this example we do it because AddIns is a logical place.
10. In `<draw_home>\AddIns`, create a file called `MyAddinTool.xml` with the following:

```
<mdldraw debug="true" popupsErrors="true">
  <tool name="Molfile Popup Tool"
        class="MyAdd_inTool.DemoTool"
        image="MyAdd_inTool.addin20-20.gif"
        index="2"
        addInAssemblyPath="Addins/MyAdd-inTool.dll"/>
</mdldraw>
```

Note:

- The Root Namespace in this case it is `MyAdd_inTool` with underscore (_), not hyphen (-).
- If your Add-in application defines its own Form, specify its owner to be the root Form of Accelrys Draw. See `MDL.Draw.Interfaces.IApp.Form` in the API Reference.

Undo incrementally for action or tool

The `MDL.Draw.Interfaces.IEditor` API enables you to define how many steps are undone when a user clicks the undo button. You do this by writing one or more "undo" blocks. Each undo block can have one or more steps to be undone. The following examples illustrate adding and testing undo blocks:

Undo for an action

This example demonstrates adding undo blocks to an add-in action. This example assumes that you have completed the steps of Add-in Action Example: DisplaySmilesStringAction.

Building the DLL for undo action

1. In Visual Studio, open `\Examples\C#\csharpexamples.sln`.
2. Open `AddIn.cs` and insert the following into the else block of the `ActionPerformed` method so that the action of adding the SMILES string is situated within undo blocks.

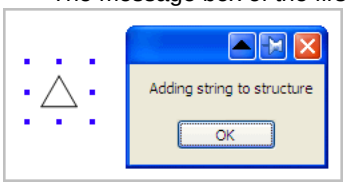
```
else
{
  string s = editor.MolfileString;

    MessageBox.Show("Adding string to structure");
    editor.BeginUndoBlock("Adding string to structure");
    // Display SMILES string
    editor.AddTextToStructure(name, this);
    editor.EndUndoBlock();

    if (s.IndexOf('C') >= 0)
    {
        MessageBox.Show("change Carbons to Nitrogens");
        editor.BeginUndoBlock("change Carbons to Nitrogens");
        // replace all C to N in molfile
        s = s.Replace("C", "N");
        // push molfile back to editor
        editor.MolfileString = s;
        editor.EndUndoBlock();
    }

    // Paste SMILES string on clipboard
    Clipboard.SetDataObject(name);
}
```

3. Compile.
4. Copy `MDL.Draw.AddIn.DisplaySmilesString.dll` from `\Examples\C#\AddInActionExample\bin\`**`Debug`** to `Examples\C#\AddInActionExample\bin\`**`Release`**.

Testing the Undo Action

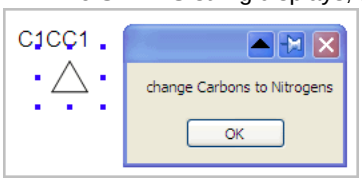
1. Start Accelrys Draw.
2. Place a structure on the canvas.
3. On the **Chemistry** menu, click **Display SMILES string**.

   The message box of the first undo block displays.



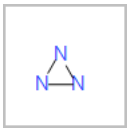
4. Click **OK**.

   The SMILES string displays, and the message box of the second undo block displays.



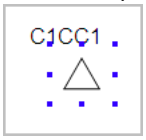
5. Click **OK**.

   The molfilestring changes, with nitrogen taking the place of carbon.



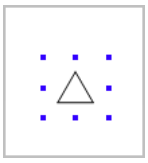
6. Click **Edit> Undo**.

   The replacement of atom types is undone.



7. Click **Undo**.

   The addition of the SMILES string is undone.



Examples: Internet Explorer

and C++

Internet Explorer Examples

This section describes the examples in the `Examples\IE\` subdirectory of your Accelrys Draw installation. These examples each consist of an HTML page with a reference to one or more external JavaScript files.

Example Name	Description	External JavaScript File(s)
dot_net_ie_renditor.htm	Gets and Sets Chime string and Molfile in a `Renditor`. See Renditor in the Internet Explorer Example.	dot_net_ie_renditor.js
dot_net_in_IE_properties.htm	Provides both a `Renderer` and a `Renditor`. *Shows how to set display preferences by using JavaScript.* See Renditor and Renderer in IE Example.	dot_net_in_IE_properties_renderer.js dot_net_in_IE_properties_renditor.js
RenditorEvent.htm	ComStructureChanged() event. See COMStructureChanged Event Example	RenditorEvent.js
RendererDoubleClicked.htm	Runs a JavaScript function when the end-user double-clicks the structure. See Demo Renderer Double Clicked Event Example	RendererDoubleClicked.js

Note: For a description of the Renderer and Renditor, see Examples Summary.

Installation Requirements for using a Renderer or Renditor in an HTML page

These installation requirements apply to any HTML page that hosts the Renderer or Renditor. Each computer that uses a rendering component on an HTML page must have a full installation of the the following products:

- Accelrys Draw for .NET
- Microsoft Internet Explorer (as the default browser)

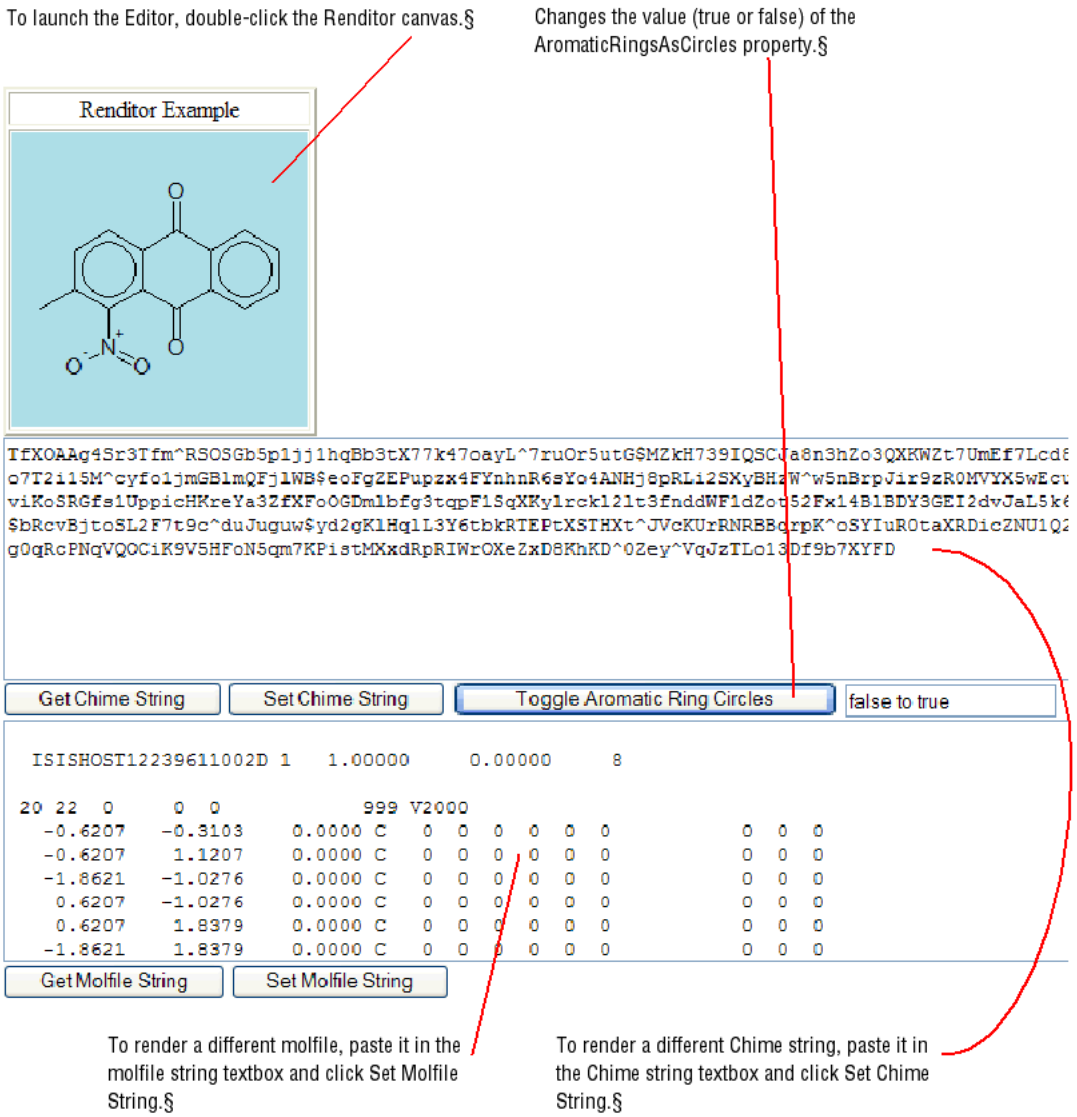
One usage scenario is for a web server application that accepts query structures from web clients and returns matching image files (by using the HeadlessRenderer) to those web clients. In such a scenario, both the web server and the clients must have an installation of Accelrys Draw.

Note: A web server with Accelrys Direct (or merely a connection to Accelrys Direct) can return chimestrings and molfiles.

Renditor in the Internet Explorer Example

Starting from the root directory of your Accelrys Draw installation, open `Examples\IE\dot_net_ie_renditor.htm`.

Internet Explorer opens displaying a page that hosts the Renditor. The Renditor canvas displays a structure.



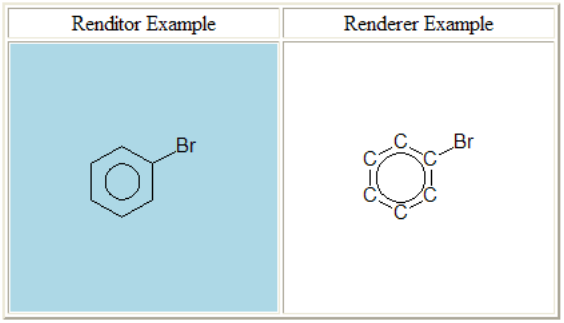
How to use the Renditor example

- To launch the Accelrys Draw Editor, double-click the canvas.
- To view the Chime string representation of the structure, click **Get Chime String.**
- To view the molfile string representation of the structure, click **Get Molfile String**.

- To display a different structure, paste in a Chime string and click the **Set Chime String** button, or paste in a molfile string and click the **Set Molfile String** button.
- To change the display preference setting for aromatic ring circles, click the **Toggle Aromatic Ring Circles** button

Renditor and Renderer in IE Example

Starting from the root directory of your Accelrys Draw installation, open `Examples\IE\dot_net_in_IE_properties.htm`. Internet Explorer opens displaying a page that hosts both a Renditor and a Renderer.



Both rendering components have the same structure, but the display preferences are different.

When you view the HTML source, you see that for the Renditor:

- Aromatic rings display as circles
- The diameter of the circle is smaller than the default of 0.7

The display preferences are set by JavaScript in the HTML source:

```
<script language="javascript">
   TestRenditor.Preferences.AromaticRingCircles = true;
   TestRenditor.Preferences.AromaticRingSize = 0.5;
</script>
```

When you view the HTML, you see that for the Renderer:

- Aromatic rings display as circles
- Carbon atoms display explicit labels

The display preferences are set by JavaScript in the HTML source:

```
<script language="javascript">
  TestRenderer.Preferences.AromaticRingCircles = true;
  TestRenderer.Preferences.CarbonLabelDisplay = true;
</script>
```

Note:

- The object tag is in a separate file, `Examples\IE\dot_net_in_IE_properties_renditor.js`
- To use a molfile, use a chime string (molfile encoded using csinline) or a `URLEncodedMolfileString`
- To use a sketch file, use SketchString (sketch encoded using base64)

About writing your external JavaScript file

The HTML page references an external JavaScript file. For example, `RendererDoubleClicked.htm` contains a reference to an external JavaScript file:

```
<script src="RendererDoubleClicked.js"></script>
```

The external JavaScript file uses the `write` method on the `document` object. The method takes an object tag as its argument.

```
document.write('<object
     id="rend"
     codebase="MDL.Draw.Editor.dll#-1,-1,-1,-1"
     height="200"
     width="200"
     classid="CLSID:FCE57399-E34B-45ce-881B-5FDFF3583307">
     <param
          name="ChimeString"
          value="[the chimestring characters here]">
     </object>')
```

When you create your own HTML page:

- You must include the `OBJECT` tag
- You can assign the value of `id` to any name you want. However, if the HTML page contains more than one rendering component (Renderer or Renditor), each must have a unique `id`. For example, `MyRenderer1`, `MyRenderer2`, and so on.
- You must use the exact `classid` and `codebase` specified in this example:

  ```
  classid="CLSID:FCE57399-E34B-45ce-881B-5FDFF3583307"
  codebase="MDL.Draw.Editor.dll#-1,-1,-1,-1"
  ```

- If you were embedding a Renderer, instead of the code above for a Renditor, you would use a different classid and codebase, as shown in `<draw_home>\Examples\IE\dot_net_in_IE_properties_renderer.js`:

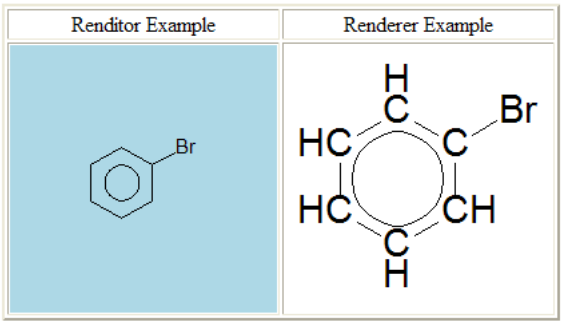
```
classid="CLSID:46803AAE-C327-4002-8CEC-05036E2FDEF4"
codebase="MDL.Draw.Renderer.dll#-1,-1,-1,-1">
```

- The syntax for the enums in the DisplayPreferences class is different when you are using Internet Explorer to host your rendering components.

```
TestRenderer.Preferences.StructureScalingMode = 1; // instead of FitToBox
TestRenderer.Preferences.HydrogenDisplayMode = 4; // instead of All
```

Note: Your HTML page must specify the value of DisplayPreference enumerations as integers. See the enumerations at `API Reference for DisplayPreferences`.

The figure below shows the effect of the settings on the structure in the Renderer.



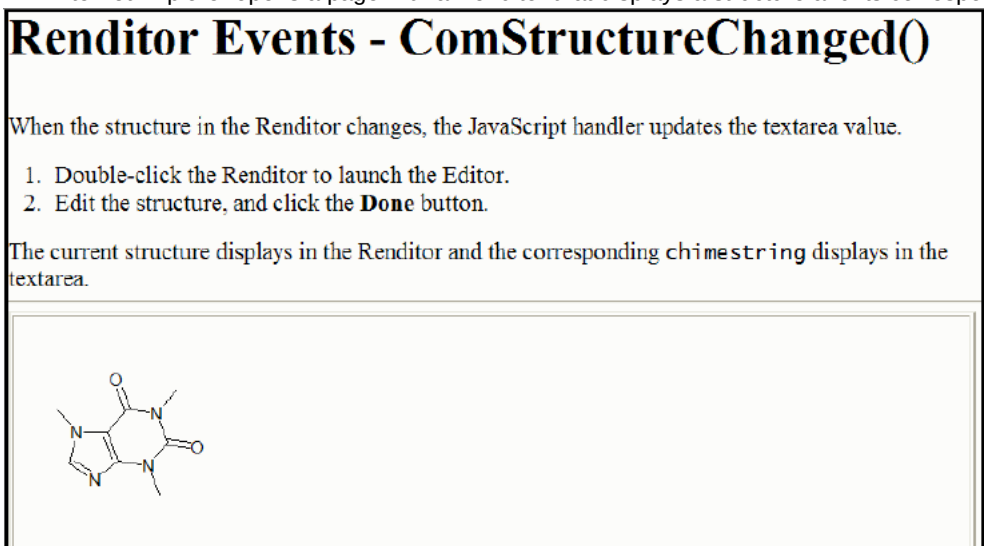
Note:

- The HTML PARAM tag is used for values that can be set without using the `DisplayPreferences` class, such as the value of `ChimeString`, `MolfileString`, `BackColor`, and `ForeColor`: `<PARAM NAME="BackColor" VALUE="LightBlue">`
- For an example of how the Renderer uses display preferences, see the Renderer class in the API Reference.
- If you set colors in the JavaScript section, the hex syntax is not RGB (Red-Green-Blue) but BGR (Blue-Green-Red). For example, the following line in the JavaScript section results in Red, not Blue.
  `TestRenditor.Preferences.BackColor = 0x0000FF;`

COMStructureChanged Event Example

Suppose that you provide an HTML page that hosts a Renditor. When an end-user modifies a structure, you can capture (sink) the `COMStructureChanged` event.

1. Starting from the root directory of your Accelrys Draw installation, open `Examples\IE\RenditorEvent.htm`. Internet Explorer opens a page with a Renditor that displays a structure and its corresponding `chimestring`.



2. Follow the on-screen instructions to cause a `COMStructureChanged` event to occur. The Renditor and the `textarea` are updated.

   The underlying JavaScript is:

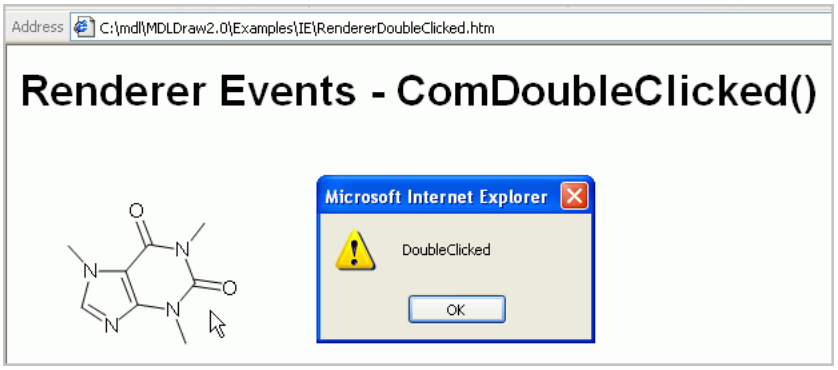
```
function rend::ComStructureChanged(){
   molstructure.value = rend.ChimeString;
   }
```

Note: See also the API Reference for the `MDL.Draw.Renditor.ComStructureChanged` delegate and the `MDL.Draw.Renditor.IRenditorEvents.ComStructureChanged` method.

Demo Renderer Double Clicked Event Example

Starting from the root directory of your Accelrys Draw installation, open `Examples\IE\RendererDoubleClicked.htm`.

This example displays a Renderer with a structure. When the double-click event occurs (because the user double-clicks the Renderer), a message box appears.



The HTML for this page (`RendererDoubleClicked.htm`) contains a reference to an external JavaScript file:

```
<script src="RendererDoubleClicked.js"></script>
```

The external JavaScript file uses the `write` method on the `document` object. The method takes an object tag as its argument.

```
document.write('<object
     id="rend"
     codebase="MDL.Draw.Editor.dll#-1,-1,-1,-1"
     height="200"
     width="200"
     classid="CLSID:FCE57399-E34B-45ce-881B-5FDFF3583307">
     <param
          name="ChimeString"
          value="[the chimestring characters here]">
     </object>')
```

ASP Example

Whereas the Renditor in the Internet Explorer Example and Renditor and Renderer in IE Example are static, the ASP example is dynamic. The server with the .asp pages responds to the client.

To support Active Server Pages (ASP), the Internet Information Service (IIS) 6.0 must be configured on the Windows 2003 Server platform.

Starting IIS

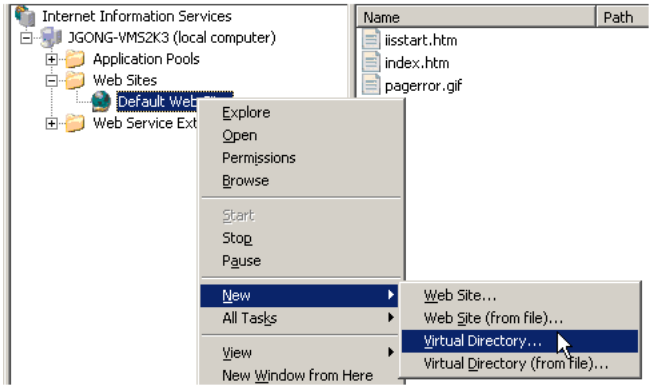
1. In Control Panel, go to Add or Remove Programs.
2. On the left side, click Add/Remove Windows Components.
   The Windows Components Wizard displays.
3. Scroll down the list of Windows Components, make sure that the Application Server checkbox is enabled, and double-click it.
4. In the Application Server dialog, make sure that the Internet Information Services (IIS) checkbox is enabled, and double-click it.
5. In the Internet Information Services (IIS) dialog, make sure that the World Wide Web Service checkbox is enabled, and double-click it.
6. In the World Wide Web Service dialog, enable the Active Server Pages checkbox.
7. Click OK three times to return to the Windows Components Wizard.
8. In the Windows Components Wizard, click Next. The Wizard configures and starts IIS.

Deploying the ASP example on the IIS server

1. In Control Panel, go to Administrative Tools, and double-click Internet Information Service (IIS).

   The Internet Information Services (IIS) Manager window displays.

2. Navigate the tree to Default Web Site. Right-click and select New > Virtual Directory.



   The Virtual Directory Wizard displays.

3. In the Virtual Directory Wizard, specify the alias for the URL the client uses to go to the default page.

   In this example, the alias is `ASP Example`.

4. In the Virtual Directory Wizard, specify the location of the `default.asp` and `result.asp` example files, which is the `Examples\ASP\` subdirectory of your Accelrys Draw installation.

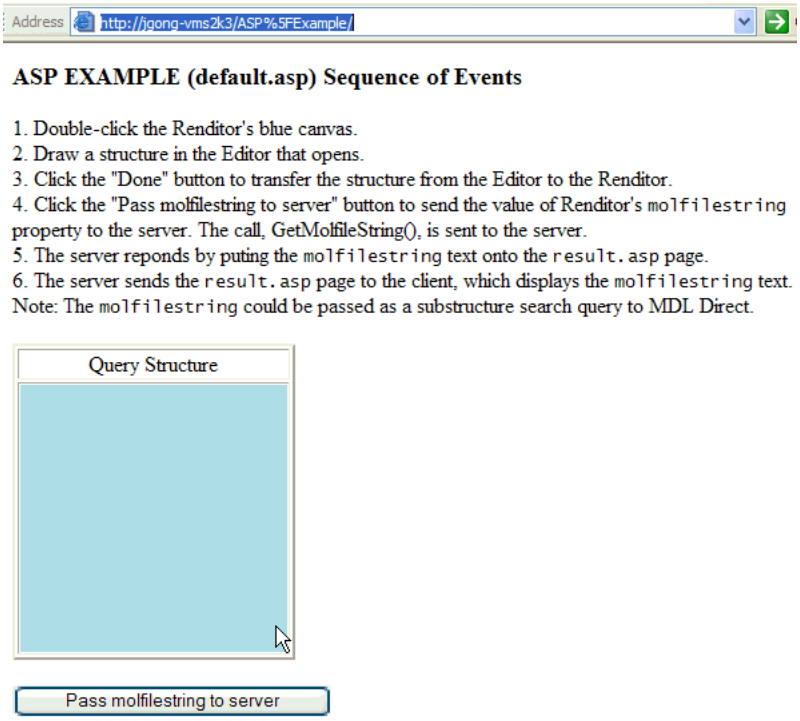
Launching the ASP through the IE client

To interact with the ASP example,

1. Start your Internet Explorer browser and go to this URL,

   `http://[server_name]/ASP%5FExample/`

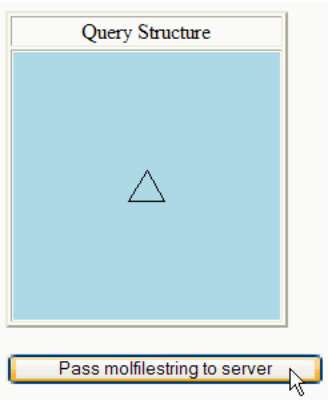
   where `server_name` is the name of the IIS server with `default.asp` as its default page, and `ASP%5FExample` is the URL encoding for the alias for the virtual directory, `ASP Example`. The default page should appear similar to what follows.



2. Double-click the Renditor canvas.

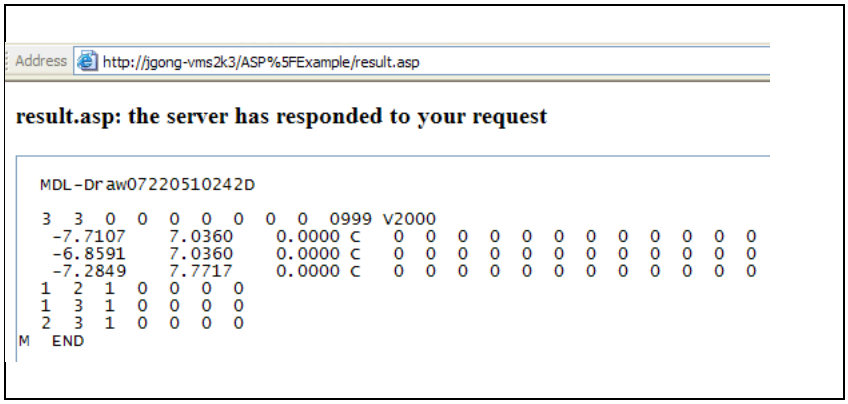
   The Accelrys Draw editor opens.

3. Draw a structure, and click the `Transfer` button to return to the default page with a structure on the canvas.



4. Click Pass molfilestring to server.

   The server sends the client a result page that displays the molfilestring.



   This concludes the test.

   For an overview of other example, see Examples Summary.

C++ Examples

These examples show how a C++ application can make use of Accelrys Draw .NET components.

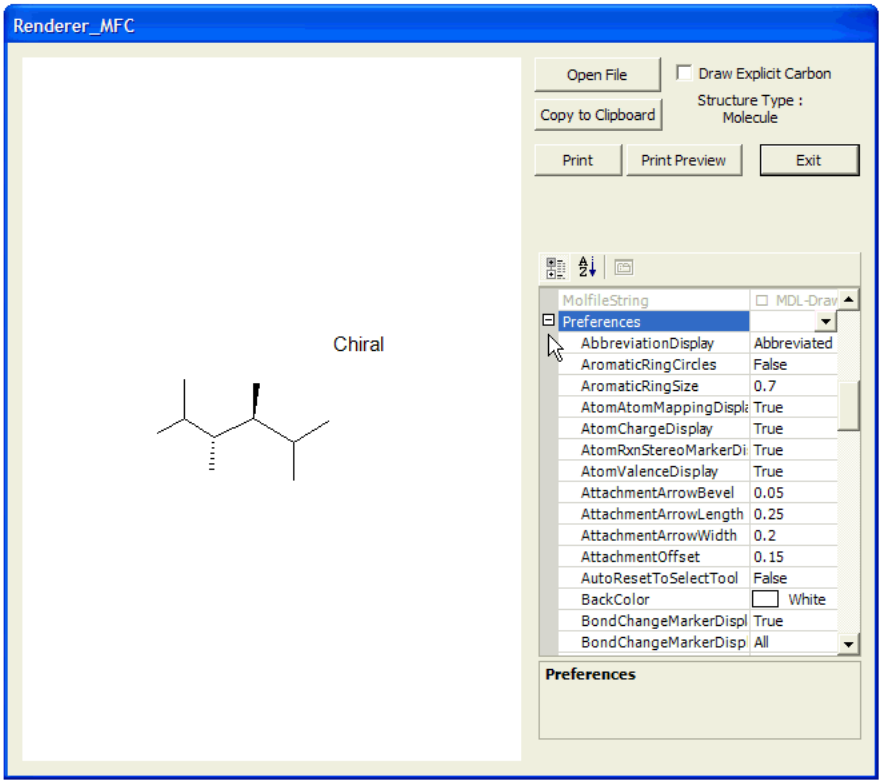
- Renderer_MFC
- Renditor_MFC

To run the C++ examples and write your own C++ application that make use of Draw's .NET components, you need both Microsoft C++ with MFC 7.0 and the software for developing .NET applications for Accelrys Draw (Microsoft Visual Studio for .NET).

Renderer_MFC

The `Renderer_MFC` example is similar to the DemoRenderer example for C# and VB.NET. To use this example, follow these steps.

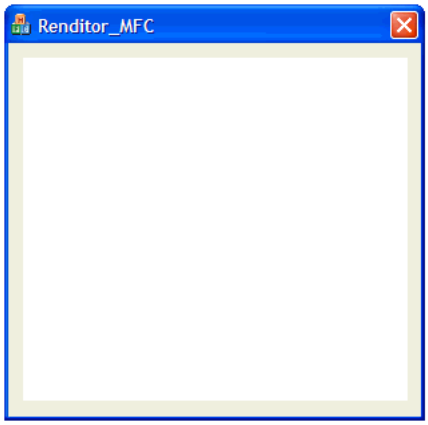
1. In the root directory of your Accelrys Draw installation, navigate to the `Examples\C++\Renderer_MFC\Release` and start `Renderer_MFC.exe`.
2. Click **Open File**, navigate to a structure file, such as a moflfile, and click **Open**.
3. Expand the **Preferences** node, and set different values for the display preferences.



Renditor_MFC

The `Renditor_MFC` example is similar to the DemoRenditorSimple example for C# and VB.NET. To use this example, follow these steps.

1. Navigate to the root directory of your Accelrys Draw installation and start `Examples\C++\Renditor_MFC\Release\Renditor_MFC.exe`.



2. Double-click the canvas to launch the Accelrys Draw Editor.

API References - Links

Accelrys Draw .NET API Reference

To access the API Reference for Accelrys Draw, see the Draw API Reference for .NET, which is available from:

- on the file system as `<draw_home>\docs\Draw\DrawNetApi.chm`
- for Draw standalone, on the Start menu for Draw > Documentation

Accelrys Draw Java API

- For the Accelrys Draw for Java developer documentation set, see the 1.2 SP1 version of Accelrys Draw.

  Note: For the javadoc for the Renditor JNI Wrapper (`JMolRendererCom`) that allows a Java application to access the .NET version of Accelrys Draw, see Draw API Reference for Renditor JNI WrapperAn example of this wrapper is documented at Java Native Interface - JNI Wrapper.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

M

N

O

P

R

S

T

U

V

W

X