Improving Developer Documentation

Contents

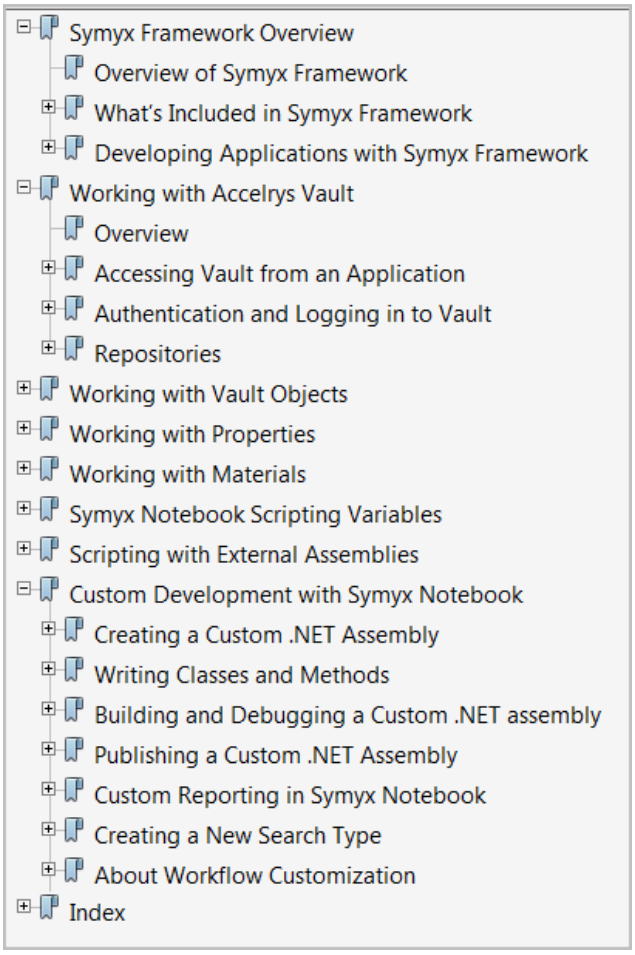
Improving the “Findability” of Developer Documentation 1

Improving the Usability of an API for a chemical drawing tool 2

Improving Navigation to Aid Code Development 7

Improving the “Findability” of Developer Documentation

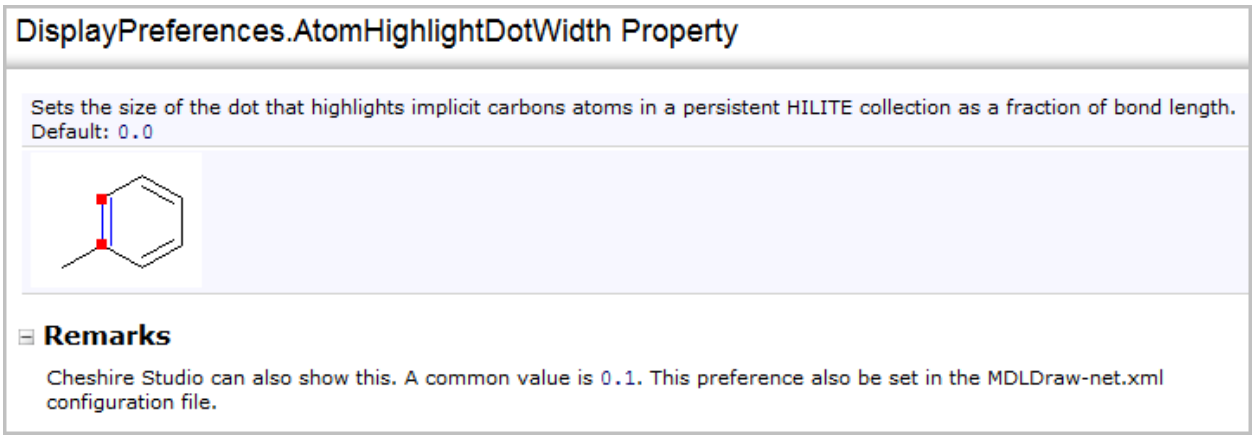
I noticed that both external customers and internal consultants asked Support for the location of various developer guides. Instead of expecting these individuals to search for the bits and pieces, I consolidated three developer guides into one guide. In addition, the consolidated will be shipped with the API Reference in a single SDK download instead of being separate.



Improving the Usability of an API for a chemical drawing tool

Instead of forcing the developer to exit the API Reference to search through the end-user Help or experiment with the graphical user interface for end-users, I integrated visuals that show how API calls change the visual representation.

The following shows the API Reference topic, followed by the C#.NET source code with the XML comments that Sandcastle Help File Builder processes into a compiled HTML (.chm) file.



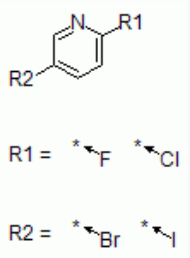

```
        // Do not highlight hidden atoms by default.
        // This replicates ISIS/Draw behavior.
        /// <summary>
        /// <table class="dtTABLE" border="0">
        /// <tr><td>Sets the size of the dot that highlights implicit carbons
atoms in a persistent HILITE collection
        /// as a fraction of bond length. Default: <c>0.0</c></td></tr>
        /// <tr><td><img
src="images/atomhighlightdotwidth.gif"></img></td></tr>
        /// </table>
        /// </summary>
        /// <remarks>Cheshire Studio can also show this. A common value is
<c>0.1</c>.
        /// This preference also be set in the MDLDraw-net.xml configuration
file.</remarks>
        [Browsable(false)] // to hide from end-user property grid!
        [DefaultValue(0.0)]
        public double AtomHighlightDotWidth {
            get { return _atomHighlightDotWidth; }
            set {
                if (_atomHighlightDotWidth != value) {
                    _atomHighlightDotWidth = value;
                    RaisePropertiesChangedEvent();
                }
            }
        } double _atomHighlightDotWidth = 0;
```

Symyx Draw 3.2 API Reference for .NET

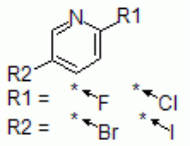
DisplayPreferences.RGMemberVerticalOffset Property

Specifies the vertical spacing between the root structure and the Rgroup definition(s) as a fraction of average bond length. Default: 0.75.

The image below shows a value of 2.0



The image below shows a value of 0.2



Remarks

- This offset spacing is between the root structure and the first definition as well as between first Rgroup definition and any other Rgroup definitions.
- The Rgroup definition consists of the Rgroup label, an equals sign (=), and one or more Rgroup member fragments.
- If you change this property, Symyx Draw renders the change when the structure is (re)loaded from file.

```
        /// <summary>
        /// <table class="dtTABLE" border="0">
        /// <tr><td>Specifies the vertical spacing between the root structure
and
        /// the Rgroup definition(s) as a fraction of average bond length.
        /// Default: <c>0.75</c>.</td></tr>
        /// <tr><td>The image below shows a value of <c>2.0</c></td></tr>
        /// <tr><td><img src="images/RGMemberVerticalOffset-
2.gif"></img></td></tr>
        /// <tr><td>The image below shows a value of <c>0.2</c></td></tr>
        /// <tr><td><img src="images/RGMemberVerticalOffset-
point2.gif"></img></td></tr>
        /// </table></summary>
        /// <remarks><ul><li>This offset spacing is between the root
structure and the first
        /// definition as well as between first Rgroup definition and any
other Rgroup
        /// definitions.</li>
        /// <li>The Rgroup definition consists of the Rgroup label,
        /// an equals sign (=), and one or more Rgroup member fragments.</li>
        /// <li>If you change this property, Symyx Draw renders the change
        /// when the structure is (re)loaded from file.</li></ul>
        /// </remarks>
```

```
        [Description("Rgroups: space between root structure and Rgroup
definition as a fraction of average bond length. Default: 0.75"),
DefaultValue(0.75)]
        public double RGMemberVerticalOffset {
            get { return _RGMemberVerticalOffset; }
            set {
                if (_RGMemberVerticalOffset != value) {
                    _RGMemberVerticalOffset = value;
                    RaisePropertiesChangedEvent();
                }
            }
        } double _RGMemberVerticalOffset = 0.75;
```

Symyx Draw 3.2 API Reference for .NET

BondChangeMarkerMode Enumeration

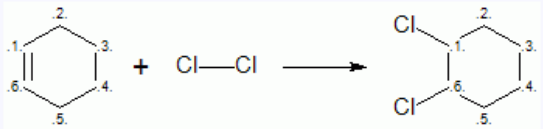
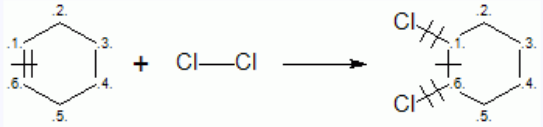
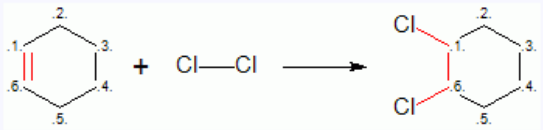
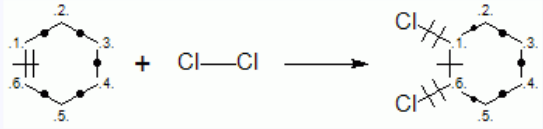
See Also Send Feedback

(Reactions): Specifies the constants to use when setting the value of BondChangeMarkerDisplayISIS.

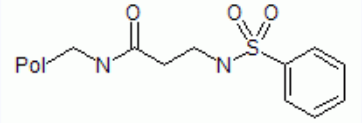
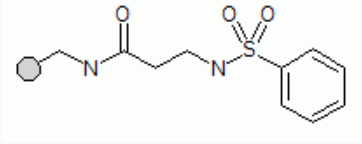
Namespace: MDL.Draw.Renderer.Preferences
Assembly: MDL.Draw.Renderer (in MDL.Draw.Renderer.dll) Version: 3.2.200.56

Syntax

Members

	Member name	Description
	Off	Displays no bond change markers. (If your Renderer is in an Internet Explorer HTML page, use 0) 
	HashMarks	Displays bond change markers solely on bonds that change in the reaction. (If your Renderer is in an Internet Explorer HTML page, use 1) 
	Color	Displays reacting centers in Red. (If your Renderer is in an Internet Explorer HTML page, use 2) 
	AllMarks	Displays bond change markers on all bonds in the reaction, including the dot marker (.) on bonds that undergo no change. (If your Renderer is in an Internet Explorer HTML page, use 3) 
	Thicker	Displays bond change markers as thicker bonds

```
    /// <summary>
    /// (Reactions): Specifies the constants to use when setting the value of
    /// <see cref="DisplayPreferences.BondChangeMarkerDisplayISIS"/>.
    /// </summary>
    public enum BondChangeMarkerMode : int {
        /// <summary>Displays no bond change markers.
        /// (If your Renderer is in an Internet Explorer HTML page,
        /// use 0)
        /// <para><img
src="images/BondChangeMarkerMode_Off.gif"></img></para>
        /// </summary>
        Off = 0,
        /// <summary>
        /// Displays bond change markers solely on bonds that change in the
reaction.
        /// (If your Renderer is in an Internet Explorer HTML page,
        /// use 1)
        /// <para><img
src="images/BondChangeMarkerMode_ChangeOnly.gif"></img></para>
        /// </summary>
        HashMarks = 1,
        /// <summary>
        /// Displays reacting centers in Red.
        /// (If your Renderer is in an Internet Explorer HTML page,
        /// use 2)
        /// <para><img
src="images/BondChangeMarkerMode_Color.gif"></img></para>
        /// </summary>
        Color = 2,
        /// <summary>
        /// Displays bond change markers on all bonds in the reaction,
        /// including the dot marker (.) on bonds that undergo no change.
        /// (If your Renderer is in an Internet Explorer HTML page,
        /// use 3)
        /// <para><img
src="images/BondChangeMarkerMode_All.gif"></img></para>
        /// </summary>
        AllMarks = 3,
        /// <summary>
        /// Displays bond change markers as thicker bonds
        /// </summary>
        Thicker = 4,
    }
```

PolBead	Specifies how to indicate the atom(s) that bind this structure to a polymer. • Default: `true`, which displays the "Pol" atoms as circles.  • If `true`, displays atoms of type `Pol` as filled circles that resemble beads.  Note: Do not confuse `Pol` atoms with the `POL` Flexmatch switch for polymers.

```
        /// <summary>
        /// Specifies how to indicate the atom(s) that bind this structure to
a polymer.
        /// <ul><li>Default: <c>true</c>, which displays the "Pol" atoms as
circles.
        /// <img src="images/pol-label.gif"></img></li>
        /// <li>If <c>true</c>, displays atoms of type
        /// <c>Pol</c> as filled circles that resemble beads.
        /// <img src="images/polbead.gif"></img>
        /// </li></ul>
        /// Note: Do not confuse <c>Pol</c> atoms with the <c>POL</c>
        /// Flexmatch switch for polymers.</summary>
        [Description("Displays atoms that bind to polymer displays as shaded
beads if true. If false, as Pol. Default: true"), DefaultValue(true)]
        public bool PolBead {
            get { return _polBead; }
            set {
                if (_polBead != value) {
                    _polBead = value;
                    RaisePropertiesChangedEvent();
                }
            }
        } bool _polBead = true;
```

Improving Navigation to Aid Code Development

After interviewing actual customers who develop scripts to enforce business rules of chemical representation, I realized that the documentation's examples are often used as starter code for the development of production code. Therefore, I added links within the examples and the descriptions to every method that is directly relevant for developing a product script that uses the method in the topic. Prior to that, only the **See also** section had links, and such links lacked the context of immediate scripting.

Avg()

For a molecule, collection, or reaction: calculates an average value for a specified property. Description of Avg()

Signature (see Syntax Conventions)

```
dbl = colMolRxn.Avg( property );
```

```
dbl = Avg( property ); // Target molecule or Target reaction
```

Object

`colMolRxn`	Collection, a molecule or reaction

Parameters

`property`	Atom property or bond property whose values you want averaged. Must be a property that has numeric values, such as B_LENGTH, A_CHARGE. See Properties used - Avg().

Returns

`dbl`	Calculated average as a double

Properties used - Avg()

The following atom and bond properties with numeric values are useful with Avg():

atoms	A_CHARGE, A_MASSREAL, A_XCOORD, A_YCOORD, A_ZCOORD
bonds	B_LENGTH

Sgroups	S_DATAXCOORD, S_DATAYCOORD

Example for Avg()

The following example uses Avg() and Append() to place a water fragment near the center (average x- and y-coordinates) of the target molecule, but with an offset in the x-axis:

```
   function AddWaterToTargetUsingAverageCoordinates() {
      molecule = CreateMol("O");
      // place the water fragment on the right side of the target molecule
      var collection = Append( molecule.All(), Max(A_XCOORD)+5,
Avg(A_YCOORD));
      // return TRUE if the water was appended
      return( collection.HasContents() );
   }
```

Description of Avg()

For a molecule or collection, `Avg()` determines an average value for a given atom or bond property. This property must be numeric. For example, to get the average bond length for the bonds in a molecule, use `mol.Avg(B_LENGTH)`.

If you use `Avg()` without any Target molecules in a Cheshire environment, it produces an error message. You can only use `Avg()` with atom or bond properties that can be assigned numeric values. If you use `Avg()` with a *non–numeric property*, such as A_SYMBOL, it produces an error message.

If you attempt to average a property that does not belong to any items in a collection, Cheshire aborts the operation and returns a failure message to the program that calls the Cheshire script. For example, if you attempt `col.ToAtoms().Avg(B_TYPE)`, Cheshire aborts the operation and returns a failure message.

`Avg()` operates on items in the Root Member and Rgroup members of a molecule.

See also

List(), ListDistinct(), Max() - for property, Min() - for property, Sum()

BondIterator()

Creates an bond Iterator object that gives you access, by means of the Next() method, to every bond in a collection, molecule, or reaction. Description of BondIterator()

Caution: The order of the objects returned from an iterator is arbitrary. Do not rely on retrieving objects in any order. The only guarantee is that the Next() method of the iterator allows you to access each object exactly once. To preserve the order of objects, create an array. For a template, see Array(): Creating an array from an iterator.

Note: An alternative to using an iterator is to use a `for/in` loop. See for/in loop with an iterator or an array.

Signatures (see Syntax Conventions)\

```
bondIt = colMolRxn.BondIterator();
```

```
bondIt = BondIterator(); // Target molecule or Target
reaction
```

Object

`colMolRxn`	Collection, molecule, or reaction

Returns

`bondIt`	Bond iterator that you can use to iterate through the bonds that a molecule, reaction, or collection contains

Example of BondIterator()

This example illustrates how you can use `BondIterator()` with a molecule.

```
function IterateThroughBonds() {
   bondIterator = BondIterator(); // Create a bond iterator.
   bond = bondIterator.Next(); // Get an arbitrary bond.
   while (bond.HasContents()) {
      // While the iterator returns bonds, perform an action.
   bond = bondIterator.Next(); // Iterate to the next bond.
   }
   return (TRUE);
}
```

These example illustrates how you can use `BondIterator()` with a molecule that contains Rgroups and Rgroup members.

```
bit = Find(M_ROLE, "R1M1").All().BondIterator();
// creates iterator that can return bonds in the member "R1M1"
bit2 = Find(R_LABEL, "R1").All().BondIterator();
// creates iterator that can return bonds in all members in Rgroup, R1
```

Description of BondIterator()

Creates a bond iterator object that you can use to iterate through the bonds in a molecule, reaction, or collection. You use the bond iterator method Next() to return a collection from the bond iterator. Each returned collection contains an individual bond in the molecule, reaction, or collection.

After a bond iterator returns every bond that it can return, it returns an empty collection. You can use the bond iterator method, Reset(), to return the first bond from the bond iterator.

If you use `BondIterator()` with a molecule, reaction, or collection that contains no bonds, `BondIterator()` creates an iterator that returns an empty collection. If you want to operate on a bond, use FirstBond().

Note:

`BondIterator()` creates an iterator object that can return the bonds in all members of a molecule
•(the Root Member and Rgroup member*s*).
•For an overview of all the Cheshire iterators, see Iterator.

See also

Iterator, Next(), Reset(), AtomIterator(), CollectionIterator(), FirstAtom(), FragmentIterator(), MapIterator(), MemberIterator(), RgroupIterator(), ProductIterator(), ReactantIterator(), RootMemberIterator(), SpecificIterator(), StereoIterator(), TransformIterator()