

SIMDIS Custom Toolbar Plug-in

1 Requirements

- [SIMDIS](#) or [Plot-XY](#) and the Custom Toolbar Plug-in, of the same compiler version. For example, for the 64-bit Windows VC-14.2 version:
 - SIMDIS/bin/amd64-nt/simdis10.exe or SIMDIS/bin/amd64-nt/PlotXY.exe
 - SIMDIS/plugins/piCustomToolbar_win64_vc-14.2.dll
- An XML file providing the layout of a custom toolbar. See [section 3](#).

2 Custom Toolbar Plug-in Information

The Custom Toolbar Plug-in for SIMDIS/Plot-XY is written using the C++ SIMDIS Plug-in API distributed by the Tactical Electronic Warfare Division at the U.S. Naval Research Laboratory. The Plug-in API allows the plug-in to seamlessly integrate with SIMDIS/Plot-XY and provide specialized utility to the end-user above and beyond the capabilities distributed with SIMDIS.

The Custom Toolbar Plug-in provides a means for end-users to customize the user interface of the application by hiding the standard SIMDIS/Plot-XY toolbars and/or creating new toolbars. User-created toolbars are defined in **XML** files and can host a number of custom buttons with various functions within the application.

Comments, issues, or feature requests regarding the Custom Toolbar Plug-in can be logged to the [SIMDIS Help Desk](#).

3 Configuring the Toolbar File

You will need to generate an **XML** (.xml) file to create a custom toolbar. Like all **XML** files, configuration values are surrounded by open tags (e.g. “<tag>”) and close tags (e.g. “</tag>”). You can write comments in your **XML** files using a pair of comment tags (“<!--” followed by “-->” to end the comment block). Custom Toolbar config files must be contained within a pair of `customToolbar` tags to be read by the plug-in:

```

<!-- This is a comment before starting the toolbar document -->
<customToolbar>
  <!--
    This is a multi-line comment inside the toolbar configuration.
    The comment won't end until the end tag is found.
  -->
</customToolbar>

```

3.1 Document Options

There is one option that can be applied to SIMDIS/Plot-XY, separate from your added toolbars:

- **hideStandardToolbars**: Omitting this option, or providing a value of “0” indicates that the standard toolbars shall not be hidden. Any other value will hide the standard toolbars before adding your custom toolbars.

All document options must be enclosed within a pair of **options** tags:

```

<customToolbar>
  <options>
    <!-- By providing a value of ‘1’ here, we hide the standard toolbars -->
    <hideStandardToolbars>1</hideStandardToolbars>
  </options>
</customToolbar>

```

4 Creating a Custom Toolbar

You can define multiple custom toolbars in a single document. The definition of each toolbar is placed within a pair of **toolbar** tags:

```

<customToolbar>
  <!-- All toolbars must be inside the customToolbar tags -->
  <toolbar>
    <!-- This is a comment inside a toolbar -->
  </toolbar>
</customToolbar>

```

4.1 Title

You can assign a Title to your toolbar, which can be used to refer to it in some GUIs:

```

<toolbar>
  <title>Custom Toolbar 1</title>
</toolbar>

```

4.2 Options

There are several options that can be applied to your toolbar:

- **iconAndTextMode**: Customizes the appearance of buttons and menus on the toolbar. Supported values are:
 - **IconAndText**: Each button or menu shows the assigned icon and text
 - **IconOnly**: Each button or menu shows only the assigned icon
 - **TextOnly**: Each button or menu shows only the assigned text
- **iconWidth**: Sets the base width of each button or menu on the toolbar.
- **iconHeight**: Sets the base height of the toolbar.

All toolbar options must be enclosed within a pair of **options** tags:

```
<toolbar>
  <options>
    <!-- Show both the icon and text of all buttons -->
    <iconAndTextMode>IconAndText</iconAndTextMode>
    <!-- Dimensions of the toolbar in pixels. Match the default size of 26x26 pixels. -->
    <iconWidth>26</iconWidth>
    <iconHeight>26</iconHeight>
  </options>
</toolbar>
```

4.3 Actions

You can add actions to your toolbar, to force some operations to occur when the toolbar is created. The full list of action names can be found in the **Hot Keys** dialog via **Tools > Hot Keys**. All actions must be enclosed within a pair of **action** tags:

```
<toolbar>
  <!-- Force the Time Editor GUI to open when the toolbar is loaded -->
  <action>Time Editor</action>
  <!-- Full-screen the application when the toolbar is loaded -->
  <action>Full Screen</action>
</toolbar>
```

4.4 Buttons

Buttons are the main element of a custom toolbar and can trigger various operations within SIMDIS/Plot-XY. Configuration options for a button are all written inside the **button** tags. There

are multiple types of button, and the button's type is specified by the opening tag. All button types share several configuration options:

- **text**: Specifies the label shown on the button (in **IconAndText** or **TextOnly** modes).
- **tooltip**: Sets the hint text to display when the user's mouse hovers over the button.
- **iconData**: Sets the button's icon to one of several built-in icons, listed in [Appendix A](#).
- **iconFile**: Sets the button's icon to one loaded from the specified file location.

NOTE: Only the final **iconData** or **iconFile** option on a button will be applied.

```
<toolbar>
  <button type="Action">
    <text>Toggle All</text>
    <tooltip>Toggle All Bars</tooltip>
    <!-- This iconData is not applied, because an iconFile tag comes afterwards -->
    <iconData>TreeExpand</iconData>
    <iconFile>$(SIMDIS_DIR)/data/icons/circle.png</iconFile>
  </button>
</toolbar>
```

4.4.1 Action Buttons

Buttons of the “Action” type can be used to trigger an action to occur in SIMDIS/Plot-XY. The full list of action names can be found in the **Hot Keys** dialog via **Tools > Hot Keys**. The name of the action to trigger must be inside **actionName** tags:

```
<toolbar>
  <button type="Action">
    <text>Toggle All</text>
    <tooltip>Toggle All Bars</tooltip>
    <iconData>TreeExpand</iconData>
    <!-- This action will toggle the display of the standard SIMDIS/Plot-XY toolbars -->
    <actionName>Toggle All Bars</actionName>
  </button>
</toolbar>
```

Action buttons linked to actions that have a toggle state will display the current toggle state of their action. See [Figure 1](#).

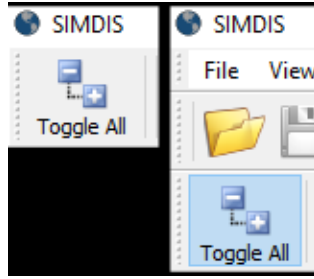


Figure 1: Button linked to the “Toggle All Bars” action, before and after activation.

4.4.2 Category Filter Buttons

Buttons of the “CategoryFilter” type can be used to show or hide entities based on their category data, using the Draw preference rule. Category Filter Buttons have several unique option tags:

Option Tag	Description	Valid Values
<code>categoryName</code>	Name of the category to check when filtering entities.	Any text string.
<code>categoryValue</code>	Category Data value to match when filtering entities.	Any text string.
<code>entityType</code>	Entity type(s) to which the Draw rule is applied. Defaults to PBGLDRC if not specified.	Valid entity type values: P=Platform, B=Beam, G=Gate, L=Laser, D=LOB(Detection), R=Projector, and C=Custom Rendering.
<code>nameExpression</code>	Regular expression, where entities with non-matching names will not be filtered. Defaults to “.*” if not specified.	Any valid regular expression.
<code>categoryFilters</code>	A more complex set of categoryFilters to which the Draw rule will be applied. The “categoryFilters” value from a SIMDIS preference rule would be entered here.	A Category Data Filter string from a SIMDIS preference rule, for example <code>Affinity(1)~Friendly(1)</code> .
<code>initialState</code>	Visible state to apply to matching entities when the toolbar is loaded.	“True” or “False”.

NOTES:

- If both the `categoryName` and `categoryValue` options are set, the `categoryFilters` option will be ignored.
- For an explanation of SIMDIS Category Data Filter strings, see the **Prefs Tool** section of the SIMDIS User Manual.

```

<toolbar>
  <button type="CategoryFilter">
    <text>Friendlies</text>
    <!-- This filter shows/hides all entities that have a 'Friendly' Affinity value -->
    <categoryName>Affinity</categoryName>
    <categoryValue>Friendly</categoryValue>
    <iconData>Gear</iconData>
  </button>
  <button type="CategoryFilter">
    <text>Alphas</text>
    <!--
      With no categoryName/categoryValue or categoryFilters options, this filter
      applies to all entities that match the name expression
    -->
    <!-- Apply this filter to entities with "Star" in their names -->
    <nameExpression>Star</nameExpression>
  </button>
  <button type="CategoryFilter">
    <text>Hostile Platforms</text>
    <!-- Apply this filter only to Platforms -->
    <entityType>P</entityType>
    <!--
      This filter string matches entities that have an Affinity value,
      if that value is Hostile
    -->
    <categoryFilters>Affinity(1)~Hostile(1)</categoryFilters>
  </button>
</toolbar>

```

When the button is pushed in, entities matching the configured filter are shown. When the button is raised, the entities will be hidden. See [Figure 2](#).

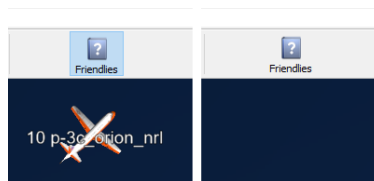


Figure 2: Category Filter Button, before and after activation.

4.4.3 DISCN Configuration Buttons

Buttons of the “DiscnConfig” type load configuration information from either a .asi or .discn file. These are useful for generating multiple display configurations. Note that only the following commands are processed in the loaded file:

Command	Description
GOGFile	Loads a GOG (.gog) file for display in the scenario.
ViewFile	Loads a View (.svml) file to configure the view.
RuleFile	Loads a Pref Rule (.rul) file to apply rules to entities in the scenario.
ITConfigFile	Loads a Map (.earth) file to configure the display of the globe.

Here is a minimal example **ASI** file, that loads a **GOG** file named “States.gog”:

```
# SIMDIS ASCII Scenario Input (ASI) File Format
Version 21
GOGFile "States.gog"
```

The name of the file to load must be contained in **discnFile** tags:

```
<toolbar>
  <button type="DiscnConfig">
    <text>GOG</text>
    <tooltip>Load the States.gog file via an ASI file</tooltip>
    <discnFile>States.asi</discnFile>
  </button>
</toolbar>
```

NOTE: If triggering a DISCN Configuration Button causes an error (e.g. the file doesn’t exist), it will be disabled until the toolbar is reloaded. You can check the application console for more information about the error.

4.4.4 Apply Rules Buttons

Buttons of the “Apply Rules” type apply one or more Preference Rules to all matching entities in the scenario. Apply Rules Buttons have two unique options:

- **rule:** Preference Rule to apply. It will be added to SIMDIS using the Plug-in API command `PISIMDIS::addPrefRule()`. The text should be escaped for XML. This option can be repeated for multiple rules on one button.
- **version:** Version for loading the rules. If omitted, then version 4 is presumed.

```

<toolbar>
  <button type="ApplyRules">
    <text>Rules</text>
    <tooltip>Apply some Preference Rules to the entities</tooltip>
    <version>4</version>
    <!-- This rule hides all "Friendly" entities, similar to the CategoryFilter example above -->
    <rule>
      ruleName=Draw ruleValue="no" nameExpression=".*" entityType=PBGLD
      categoryFilters="Affinity(1)~Friendly(1)"
    </rule>
  </button>
</toolbar>

```

NOTE: Unlike Category Filter Buttons, Apply Rules Buttons are do not have an “undo” state. Clicking the button a second time will re-apply the same rule.

4.5 Separators

Separators are small vertical bars, and can be used to group buttons on your toolbars. A separator is added using a single “separator” tag:

```

<toolbar>
  <button type="Action">
    <text>Button 1</text>
    <actionName>Toggle All Bars</actionName>
  </button>
  <!-- Separator is created by a single tag, note the '/' before the ending '>' -->
  <separator/>
  <button>
    <text>Button 2</text>
    <actionName>Open</actionName>
  </button>
</toolbar>

```

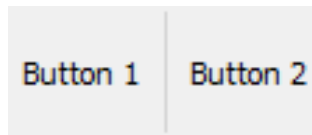


Figure 3: Two buttons, with a separator between.

4.6 Menus

Menus allow multiple buttons and separators to be grouped together on the toolbar, only displayed when the menu is opened. Menus support the `text`, `tooltip`, `iconData`, and `iconFile` options much the same as buttons do. Those options and any elements you want inside the menu must all be defined within the `menu` tags:


```

<toolbar>
  <menu>
    <text>Menu</text>
    <iconData>Gear</iconData>
    <!-- Define the first button inside the menu -->
    <button type="Action">
      <text>Time Editor</text>
      <iconData>Clock</iconData>
      <actionName>Time Editor</actionName>
    </button>
    <separator/>
    <button type="Action">
      <text>Toggle Labels</text>
      <iconData>IncreaseFont</iconData>
      <actionName>Platform Labels</actionName>
    </button>
  </menu>
</toolbar>

```

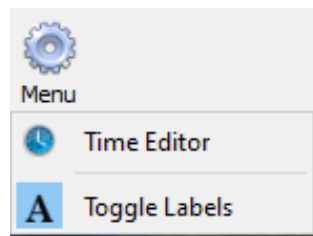















Figure 4: An opened menu, with two buttons and a separator inside.

A Built-in Icons

The following icons are available for use by Custom Toolbar **XML** files:

Icon	iconData Name	Description
	Add	The addition sign (a plus).
	ArrowDown	An arrow pointing down.
	Camera	A tool used to capture video.
	Clock	A round, analog clock.
	Edit	A pencil.
	Export	A dashed arrow, pointing up.
	Gear	A round cog, with teeth.
	Help	A question mark on a book.
	IncreaseFont	A large, capital letter 'A'.
	Load	Typical icon for opening a file.
	TapeMeasure	A tool used to measure distance.
	Texas	The state with surrounding area.
	TreeExpand	Two folders connected by lines with a '+/-'.

NOTE: If the `iconFile` option is used but the listed icon file cannot be found, then an error icon will be used instead: 