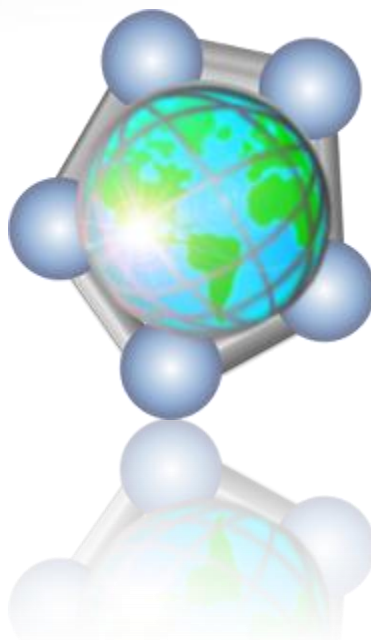




Gaia 3.4

Extenders Developer's Guide

Version 3.4.0.0013



Carbon Project, Inc.
25 Mall Road – Suite 300
Burlington, MA 01803

info@TheCarbonProject.com
www.TheCarbonProject.com



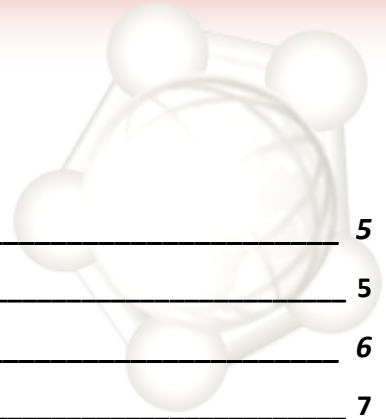
Copyright Message

Information in this document, including URLs and other Internet Web site references, is subject to change without notice. The Carbon Project may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from The Carbon Project, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

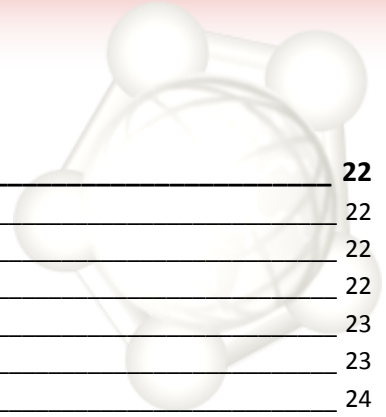
© 2004-2009 Carbon Project, Inc. All rights reserved.

The Carbon Project, CarbonTools, CarbonTools PRO, CarbonCloud and Geosocial Networking are trademarks or registered trademarks of Carbon Project, Inc.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.



About The Carbon Project	5
About CarbonTools PRO	5
Overview	6
Simple Extender Code	7
Extenders API Methods and properties	8
Legend	8
Syntax	8
General Purpose	9
Description	9
GaiaVersion	9
SetExtendersImplementers	10
Name	10
User Interface Elements	11
Setting Extender Controls in Gaia	11
GetBottomToolStripPanelTools	11
GetLeftToolStripPanelTools	12
GetMenuStripItems	13
GetRightToolStripPanelTools	14
GetStatusStripTools	15
GetTopToolStripPanelTools	16
MapBackgroundColor	17
OverridingSplashImage	17
Getting Gaia's Controls	18
SetLayersPanel	18
SetLegendPanel	18
SetMainForm	19
SetMainMapPanel	19
SetMenuStrip	20
SetStatusStrip	20
SetToolStripContainer	21



CarbonTools PRO Based Calls	22
Getting Gaia's CarbonTools based Controls	22
SetLayersContainer	22
SetMultiMap	22
SetNotesContainer	23
SetNotesSymbolSelectorDialog	23
SetSymbolSelectorDialog	24
Modifying the Layer Info Tool	25
GetFeaturesInfoPanel	25
GetLayerInfoPanel	26
Modifying the Layer Properties View	27
AcceptLayerProperty	27
AcceptLayerPropertyFinal	27
GetLayerPropertyCustomPages	28
GetLayerPropertyPanelNames	28
Events	29
Event Handlers	29
GaiaToExtenderEventHandler	29
ExtenderToGaiaEventHandler	29
Gaia Map Events	30
SetMouseLocation	30
SetMapBBox	30
SessionLoaded	31
Triggering Gaia Functionality	32
OnAddFile	32
OnAddGeoObject	32
OnChangeCenterMap	33
OnChangeMapBBox	33
OnLoadInsertSession	34
OnLoadSession	34
OnMarkSessionAsUnsaved	35
OnRefreshAllLayers	35
OnRefreshLayer	36
OnRemoveGeoObject	36
OnStartNewSession	37

About The Carbon Project

The Carbon Project® (www.TheCarbonProject.com) is an innovative, high-energy software and technology company, specializing in mapping, geospatial interoperability and geosocial solutions. Our company serves geospatial professionals, software developers, government agencies and businesses that develop mapping solutions or use geospatial data from many sources.

The Carbon Project develops mapping software solutions for government and commercial purposes. Our core clientele is in the Defense and Intelligence, Environmental and Infrastructure market sectors. Our customers include the US Department of Defense (DoD), Army Corps of Engineers, UK Ministry of Defence (UK MOD), National Geospatial Intelligence Agency (NGA), Joint Forces Command, United States Geological Survey (USGS), British Geological Survey, Canada's GeoConnections, defense integrators including BAE Systems, SRA International, and Lockheed Martin and more.

Our core suite of products includes CarbonTools™ PRO, CarbonArc® PRO, and Gaia® Extenders. These software products are provided 'off-the-shelf' to customers and users around the world.

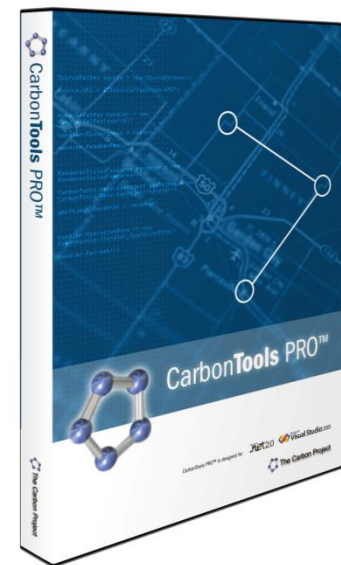
About CarbonTools PRO

CarbonTools PRO (www.CarbonTools.com) is an extension to the Microsoft .NET Framework that supports advanced location content handling and sharing and is the foundation for the Gaia application.

Based on the Source-Handler-Data® technology, CarbonTools PRO provides a unified framework for geospatial interoperability with an array of location content and services. This means Microsoft Bing Maps, Google Earth, OpenStreetMap, Yahoo! Maps, OGC SDI, GML, GMLsf, ESRI Shapefiles and more can be used in your open-geospatial .NET applications...seamlessly!

With CarbonTools PRO, .NET developers can extend existing geospatial systems, including but not limited to Gaia, with new capabilities and content sources or build exciting new open-geospatial .NET applications. Furthermore, the package contains numerous code samples and extensive documentation, including the *complete project and source code of the latest Gaia version*.

CarbonTools PRO is now available and includes many new features to support your open-geospatial .NET development projects.



Overview

Gaia® is an open-geospatial viewer application. Based on the **CarbonTools PRO** open-geospatial development toolkit (www.CarbonTools.com), this viewer can access an array of geospatial sources such as the *Open Geospatial Consortium* (OGC) Web Mapping Service (WMS), Web Coverage Service (WCS), and Web Feature Service (WFS), commercial services such as *Microsoft Bing Maps*, *OpenStreetMap* and *Yahoo! Maps* as well as various file formats including *ESRI Shapefiles*, *Google Earth KML/KMZ*, DXF, MIF, Geography Markup Language (GML) and GML for Simple Features (GMLsf). Beginning with version 3.2 (released in mid 2008) an open Extenders API was introduced. Commercial entities, government agencies and individuals can now extend and expand Gaia with new functionalities and capabilities. Some of the API functions do not require a CarbonTools PRO developer license, others do (these will be clearly marked in this document).

The Gaia Extenders system is using the .NET reflection mechanism to define a contract between the main Gaia application and any assembly (DLL) present at the Extenders designated directory. This is a non intrusive process that does not require component registration of any kind. Once Gaia detects an assembly that implements a class named **ExtenderImplementer** the assembly is considered an Extender and will be dynamically loaded and used. Once the Extender is read it will be listed in the Extenders management tool. Through the .NET reflection mechanism Gaia will look for specific API commands by name and type and implement their functionality as needed. The user may elect to disregard an Extender by un-checking it in the Extenders management form. This will prevent the corresponding assembly from being loaded by Gaia in future sessions.

Extenders can perform a variety of functions within Gaia. They can define new data types to be used by Gaia. They can construct new tools that are activated from added toolbars and controls. Extenders can also alter Gaia's look and feel by adding new user interface components or changing the behavior of existing ones.

Due to the multi-threaded nature of Gaia, where many functions are performed a-synchronously, some external operations must be addressed via event handlers. To ease the interaction with Gaia and ensure coherency and consistency of the various data types and event handlers a shared assembly named **Gaia.ExtenderHelper.dll** is available and should be referenced by the Extender. This assembly defines two event handlers and related data structures. One handler is used for events generated by Gaia and captured by the Extender, for example a change in the viewed world coordinates. This event handler can be used to perform functions related to raised events using internal callbacks. The second handler is for events that are raised by the Extender and are captured and handled by Gaia, for example centering the map or adding a new layer.

Simple Extender Code

This C# code sample shows how a very basic extender should be implemented. Compiling an assembly containing this class and placing the resulting DLL in the Gaia directory, where Gaia.exe is found, will result in this DLL being loaded as an extender, showing the specified name and description in the form opened via the File→Extenders menu command. Note there are no restrictions on the namespace name as Gaia looks for the **ExtenderImplementer** class by name alone, disregarding the namespace. It is important to note that ExtenderImplementer should be used only once per extender.

```
namespace GaiaExtender_Sample
{
    class ExtenderImplementer
    {
        public string Name
        {
            get { return "My Extender"; }
        }

        public string Description
        {
            get { return "This extender does something"; }
        }
    }
}
```

Extenders API Methods and properties



Legend

A property is marked with a brown background for the table header:

Property Name

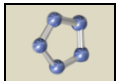
A method is marked with a dark-blue background for the table header:

Method Name

An event is marked with a dark-purple background for the table header and the title shows the handler supporting the event and the event name separated with a dot:

Handler.Event Name

Properties or methods that require a CarbonTools PRO developer license are marked with The Carbon Project logo (for more information about CarbonTools PRO please visit: www.CarbonTools.com):



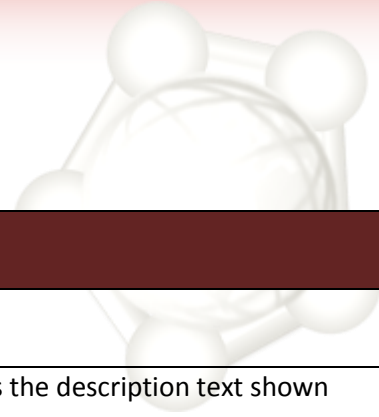
Code samples appear in Visual Studio color scheme on a light-gray background:

```
public string Name
{
    get { return "My Extender"; }
}
```

Syntax

Note that the functions names are from Gaia's perspective so a method that begins with "Set" means that Gaia sets a value for the Extender to read. A "Get" type method means the Extender is sending something to Gaia. Methods that start with "On" are meant to activate certain Gaia functionality via an associated event.

General Purpose



Description

Expected return types	string
-----------------------	--------

A property that gets the description for the Extender. This text appears as the description text shown when viewing the extenders list form in Gaia.

```
public string Description
{
    get { return "This extender does something"; }
}
```

GaiaVersion

Expected type	string
---------------	--------

Gaia populates this property with its current assembly version. This value is useful when deploying extenders that implement functionality (or extender API) that is valid from a certain Gaia version, and will not work on a lower version. The extender can verify the Gaia version and perform its functionality accordingly, e.g. pop an error message.

Gaia version consists of a dot-separated major, minor and build version (e.g. "3.2.2").

```
public string GaiaVersion { get; set; }

// Return true if the Gaia version is equal or higher than 3.2.2
bool VerifyGaiaVersion()
{
    if (String.IsNullOrEmpty(GaiaVersion)) return false;

    string[] verParts = GaiaVersion.Split(new char[] { '.' });
    if (verParts.Length != 4) return false;
    int v1 = int.Parse(verParts[0]);
    if (v1 > 3) return true;
    if (v1 < 3) return false;

    int v2 = int.Parse(verParts[1]);
    if (v2 > 2) return true;
    if (v2 < 2) return false;

    int v3 = int.Parse(verParts[2]);
    if (v3 >= 2) return true;
    return false;
}
```

SetExtendersImplementers

Parameters	Dictionary<string, object[]> extenders
Expected return types	none

Returns Gaia's list of loaded extenders as a dictionary with the assemblies' names as the key and an object array. The object array value always has the size of 2 where the first value is the Extender Instance object and the second is the ExtenderImplementer Type retrieved by Gaia.

This method is used to create dynamic linkage between individual Extenders. For example, a geometry snapping tool can be referenced and used by various Extenders using feature digitizing tools.

```
public void SetExtendersImplementers(Dictionary<string, object[]> extenders)
{
    if (extenders.ContainsKey("GaiaExtender_GeometryEdit_SnapTools"))
    {
        object[] o = extenders["GaiaExtender_GeometryEdit_SnapTools"];
        GlobalData.SnapExtenderInstance = o[0];
        GlobalData.SnapExtenderImplementer = o[1] as Type;
    }
}

// Enable or disable the 'Snap to Any' button on the snap Extender
public static void SetSnapToAny(bool enabled)
{
    GlobalData.SnapExtenderImplementer.InvokeMember("SnapToAllEnabled",
        BindingFlags.Default | BindingFlags.SetProperty | BindingFlags.Instance |
        BindingFlags.Public, null, GlobalData.SnapExtenderInstance,
        new object[] { enabled });
}
```

Name

Expected type	string
---------------	--------

A property that gets the name for the Extender. This text appears as the name of the module in the extenders list form shown by Gaia.

```
public string Name
{
    get { return "My Extender"; }
}
```

User Interface Elements

Setting Extender Controls in Gaia



GetBottomToolStripPanelTools

Parameters

none

Expected return types

System.Windows.Forms.ToolStrip
List< System.Windows.Forms.ToolStripItem>

Add tools to the bottom tool strip

```
ToolStripButton toolStripButtonNew;
ToolStrip toolStripNew;

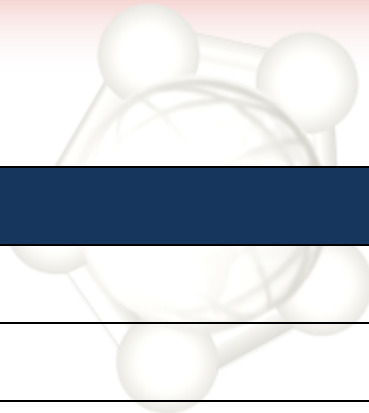
public ToolStrip GetBottomToolStripPanelTools ()
{
    this.toolStripNew = new ToolStrip();
    this.toolStripButtonNew = new ToolStripButton();

    // Add the button to the tool strip
    this.toolStripNew.Items.AddRange(new ToolStripItem[]
        { this.toolStripButtonNew });

    // Set the new button
    this.toolStripButtonNew.Image = Bitmap.FromFile("MyNewButton.bmp");
    this.toolStripButtonNew.Size = new System.Drawing.Size(22, 22);
    this.toolStripButtonNew.ToolTipText = "This is my new tool";
    this.toolStripButtonNew.Click +=
        new System.EventHandler(toolStripButtonNew_Click);

    return toolStripNew;
}

void toolStripButtonNew_Click(object sender, EventArgs e)
{
    // Do something new
}
```



GetLeftToolStripPanelTools

Parameters	none
Expected return types	System.Windows.Forms.ToolStrip List< System.Windows.Forms.ToolStrip>

Add tools to the left tool strip

```
ToolStripButton toolStripButtonNew;
ToolStrip toolStripNew;

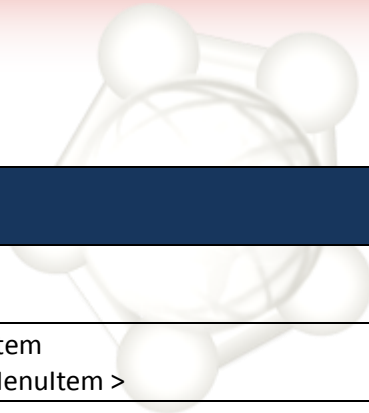
public ToolStrip GetLeftToolStripPanelTools()
{
    this.toolStripNew = new ToolStrip();
    this.toolStripButtonNew = new ToolStripButton();

    // Add the button to the tool strip
    this.toolStripNew.Items.AddRange(new ToolStripItem[]
        { this.toolStripButtonNew });

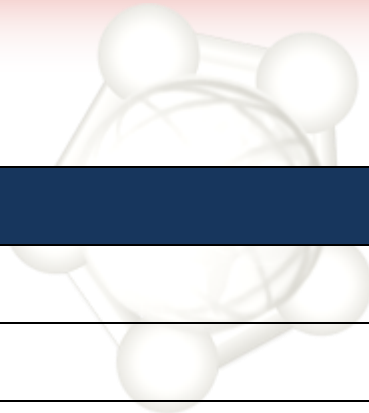
    // Set the new button
    this.toolStripButtonNew.Image = Bitmap.FromFile("MyNewButton.bmp");
    this.toolStripButtonNew.Size = new System.Drawing.Size(22, 22);
    this.toolStripButtonNew.ToolTipText = "This is my new tool";
    this.toolStripButtonNew.Click +=
        new System.EventHandler(toolStripButtonNew_Click);

    return toolStripNew;
}

void toolStripButtonNew_Click(object sender, System.EventArgs e)
{
    // Do something new
}
```



GetMenuStripItems	
Parameters	none
Expected return types	System.Windows.Forms.ToolStripMenuItem List< System.Windows.Forms.ToolStripMenuItem >
<p>Add menu item to the main menu bar. Note that the new items will be added before the “Help” menu item (which always shows last).</p>	
<pre>ToolStripMenuItem item1ToolStripMenuItem; ToolStripMenuItem item2ToolStripMenuItem; public ToolStripMenuItem GetMenuStripItems() { ToolStripMenuItem newToolStripMenuItem = new ToolStripMenuItem("New Menu Item"); item1ToolStripMenuItem = new ToolStripMenuItem("Option 1"); item2ToolStripMenuItem = new ToolStripMenuItem("Option 2"); newToolStripMenuItem.DropDownItems.AddRange(new ToolStripItem[] { item1ToolStripMenuItem, item2ToolStripMenuItem }); item1ToolStripMenuItem.CheckOnClick = true; item2ToolStripMenuItem.CheckOnClick = true; item1ToolStripMenuItem.CheckedChanged += new EventHandler(item1ToolStripMenuItem_CheckedChanged); item2ToolStripMenuItem.CheckedChanged += new EventHandler(item2ToolStripMenuItem_CheckedChanged); return newToolStripMenuItem; }</pre>	



GetRightToolStripPanelTools

Parameters

none

Expected return types

System.Windows.Forms.ToolStrip
List< System.Windows.Forms.ToolStrip>

Add tools to the right tool strip

```
ToolStripButton toolStripButtonNew;  
ToolStrip toolStripNew;  
  
public ToolStrip GetRightToolStripPanelTools()  
{  
    this.toolStripNew = new ToolStrip();  
    this.toolStripButtonNew = new ToolStripButton();  
  
    // Add the button to the tool strip  
    this.toolStripNew.Items.AddRange(new ToolStripItem[]  
        { this.toolStripButtonNew });  
  
    // Set the new button  
    this.toolStripButtonNew.Image = Bitmap.FromFile("MyNewButton.bmp");  
    this.toolStripButtonNew.Size = new System.Drawing.Size(22, 22);  
    this.toolStripButtonNew.ToolTipText = "This is my new tool";  
    this.toolStripButtonNew.Click +=  
        new System.EventHandler(toolStripButtonNew_Click);  
  
    return toolStripNew;  
}  
  
void toolStripButtonNew_Click(object sender, System.EventArgs e)  
{  
    // Do something new  
}
```

GetStatusStripTools

Parameters

none

Expected return types

System.Windows.Forms.ToolStripItem
List< System.Windows.Forms.ToolStripItem>

Add tools to the status strip

```
ToolStripStatusLabel toolStripNew;
private System.Windows.Forms.ToolStripDropDownButton toolStripDropDownButton1;
private System.Windows.Forms.ToolStripItem item1ToolStripMenuItem;
private System.Windows.Forms.ToolStripItem item2ToolStripMenuItem;

// Add a tool strip with text and a drop-down button to the Gaia status strip
public List<ToolStripItem> GetStatusStripTools()
{
    this.toolStripNew = new ToolStripStatusLabel();
    this.toolStripDropDownButton1 = new ToolStripDropDownButton();
    this.item2ToolStripMenuItem = new ToolStripMenuItem();
    this.item1ToolStripMenuItem = new ToolStripMenuItem();

    this.toolStripNew.BorderStyle = Border3DStyle.Flat;
    this.toolStripNew.Visible = true;
    this.toolStripNew.Text = "New status strip item";

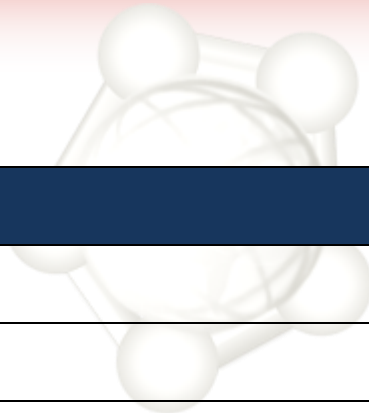
    this.toolStripDropDownButton1.DisplayStyle = ToolStripItemDisplayStyle.Image;
    this.toolStripDropDownButton1.DropDownItems.AddRange(
        new System.Windows.Forms.ToolStripItem[] {
            this.item1ToolStripMenuItem, this.item2ToolStripMenuItem});
    this.toolStripDropDownButton1.Text = "";
    this.toolStripDropDownButton1.Name = "toolStripDropDownButton1";
    this.toolStripDropDownButton1.Size = new System.Drawing.Size(45, 20);
    this.toolStripDropDownButton1.Text = "toolStripDropDownButton1";

    this.item2ToolStripMenuItem.Name = "item1ToolStripMenuItem";
    this.item2ToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
    this.item2ToolStripMenuItem.Text = "Option 1";
    this.item2ToolStripMenuItem.Click +=
        new System.EventHandler(this.item1ToolStripMenuItem_Click);

    this.item1ToolStripMenuItem.Name = "item2ToolStripMenuItem";
    this.item1ToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
    this.item1ToolStripMenuItem.Text = "Option 2";
    this.item1ToolStripMenuItem.Click +=
        new System.EventHandler(this.item2ToolStripMenuItem_Click);

    List<ToolStripItem> tools = new List<ToolStripItem>(2);
    tools.Add(this.toolStripNew);
    tools.Add(toolStripDropDownButton1);

    return tools;
}
```



GetTopToolStripPanelTools

Parameters

none

Expected return types

System.Windows.Forms.ToolStrip
List< System.Windows.Forms.ToolStrip>

Add tools to the top tool strip

```
ToolStripButton toolStripButtonNew;
ToolStrip toolStripNew;

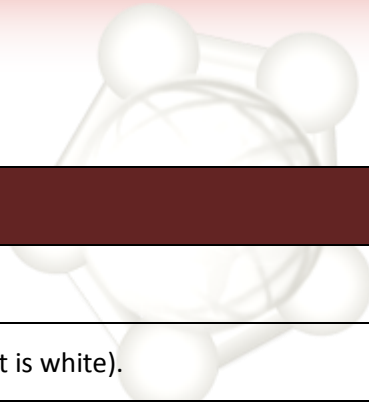
public ToolStrip GetTopToolStripPanelTools()
{
    this.toolStripNew = new ToolStrip();
    this.toolStripButtonNew = new ToolStripButton();

    // Add the button to the tool strip
    this.toolStripNew.Items.AddRange(new ToolStripItem[]
        { this.toolStripButtonNew });

    // Set the new button
    this.toolStripButtonNew.Image = Bitmap.FromFile("MyNewButton.bmp");
    this.toolStripButtonNew.Size = new System.Drawing.Size(22, 22);
    this.toolStripButtonNew.ToolTipText = "This is my new tool";
    this.toolStripButtonNew.Click +=
        new System.EventHandler(toolStripButtonNew_Click);

    return toolStripNew;
}

void toolStripButtonNew_Click(object sender, System.EventArgs e)
{
    // Do something new
}
```

MapBackgroundColor

Expected type	System.Drawing.Color
---------------	----------------------

Sets the Gaia map's default map surface background color (existing default is white).

```
public Color MapBackgroundColor
{
    get { return Color.Black; }
}
```

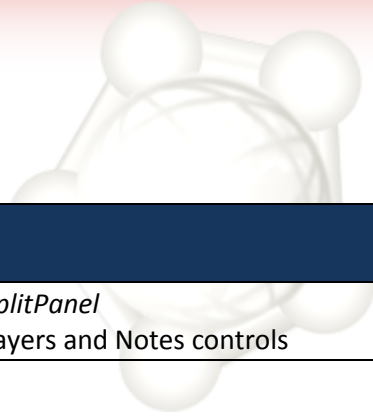
OverridingSplashImage

Expected type	System.Drawing.Image
---------------	----------------------

Sets an image that replaces the splash image that appears on the map control if no layers exist in the map. Notice that only one extender can override the splash image, the first extender loaded by Gaia that implements the *OverridingSplashImage* will be used. Any additional extenders implementing the property will be ignored. The extender may set to **null** to use a blank image.

```
public Image OverridingSplashImage
{
    get { return Bitmap.FromFile("MyNewGaiaSplash.PNG"); }
}
```

Getting Gaia's Controls



SetLayersPanel

Parameters	<i>System.Windows.Forms.SplitContainer splitPanel</i> The split container control holding the Layers and Notes controls
------------	--

Expected return types	None
-----------------------	------

Sets Gaia's side **SplitContainer** control that holds the layers and notes panels. This panel is docked to the left of the map control by default. This panel's visibility can be changed by the user using the "View" menu items.

```
Panel gaiaLayersPanel;  
  
public void SetLegendPanel(Panel gaiaLayersPanel)  
{ this.gaiaLayersPanel = gaiaLayersPanel; }
```

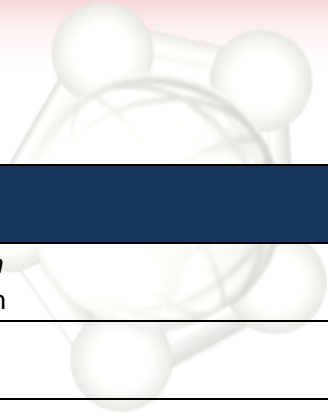
SetLegendPanel

Parameters	<i>System.Windows.Forms.Panel panel</i> The Panel control containing the Legend control
------------	--

Expected return types	none
-----------------------	------

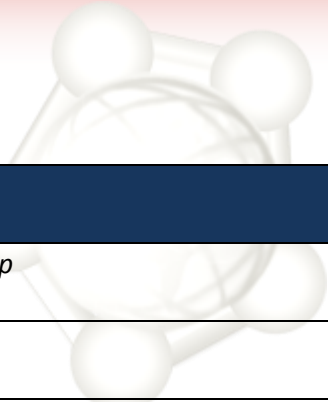
Sets Gaia's side **Panel** that contains the Legend. Originally this panel is docked to the right of the map view. This panel's visibility can be changed by the user using the "View" menu items.

```
Panel gaiaLegendPanel;  
  
public void SetLegendPanel(Panel gaiaLegendPanel)  
{ this.gaiaLegendPanel = gaiaLegendPanel; }
```



SetMainForm	
Parameters	<i>System.Windows.Forms.Form gaiaMainForm</i> The main Form object of the Gaia application
Expected return types	none
Sets Gaia's main Form object.	
<pre>// Add a side panel to the Gaia main form public void SetMainForm(Form gaiaMain) { gaiaMain = gaiaMain; gaiaMain.Controls.Add(this.sidePanel); this.sidePanel.SendToBack(); }</pre>	

SetMainMapPanel	
Parameters	<i>System.Windows.Forms.Panel panel</i> The Panel control containing the main ToolStripContainer and map
Expected return types	none
Sets Gaia's Panel that contains the ToolStripContainer around the map control.	
<pre>Panel gaiaMainMapPanel; public void SetMainMapPanel(Panel gaiaMainMapPanel) { this.gaiaMainMapPanel = gaiaMainMapPanel; }</pre>	



SetMenuStrip	
Parameters	<i>System.Windows.Forms.MenuStrip menuStrip</i> The MenuStrip control in Gaia
Expected return types	none
Sets Gaia's StatusStrip that contains all the tools in the status bar. The returned value contains all tools, including those pushed by other extenders (e.g. using GetStatusStripTools).	
<pre>ToolStripMenuItem newConfigToolStripMenuItem; // Add a new separator and item to Gaia's Tools->Configuration menu item public void SetMenuStrip(MenuStrip gaiaMenuStrip) { ToolStripItem[] configs = gaiaMenuStrip.Items.Find("configurationToolStripMenuItem", true); if (configs.Length == 0 !(configs[0] is ToolStripMenuItem)) return; this.newConfigToolStripMenuItem = new ToolStripMenuItem("My new config", null, new System.EventHandler(this.newConfigToolStripMenuItem_Click), "newConfigToolStripMenuItem"); ((ToolStripMenuItem) configs[0]).DropDownItems.Add(new ToolStripSeparator()); ((ToolStripMenuItem) configs[0]).DropDownItems.Add(newConfigToolStripMenuItem); }</pre>	

SetStatusStrip	
Parameters	<i>System.Windows.Forms.StatusStrip statusStrip</i> The StatusStrip control in Gaia
Expected return types	none
Sets Gaia's StatusStrip that contains all the tools in the status bar. The returned value contains all tools, including those pushed by other extenders (e.g. using GetStatusStripTools).	
<pre>ToolStripStatusLabel gaiaStatusLabel; public void SetStatusStrip(StatusStrip gaiaStatusStrip) { gaiaStatusLabel = gaiaStatusStrip.Items[0] as ToolStripStatusLabel; }</pre>	

SetToolStripContainer

Parameters

System.Windows.Forms.ToolStripContainer toolStrip
The ToolStripContainer control surrounding the map view in Gaia

Expected return types

none

Sets Gaia's **ToolStripContainer** that contains all the tool bars around the map control. The returned value contains all tools, including those pushed by other extenders (e.g. using **GetTopToolStripPanelTools**)

```
ToolStripButton toolStripButtonNew;
ToolStrip toolStripNew;
// Add a tooltip to to Gaia's top tool-strip panel
public void SetToolStripContainer(ToolStripContainer gaiaToolsStripContainer)
{
    this.toolStripNew = new ToolStrip();
    this.toolStripButtonNew = new ToolStripButton();

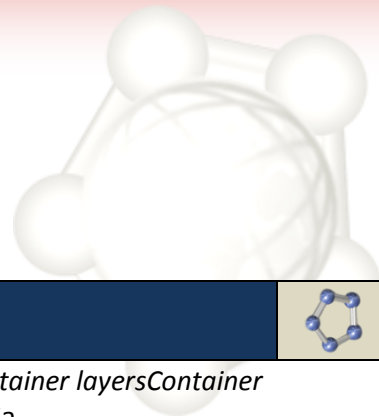
    // Add the button to the tool strip
    this.toolStripNew.Items.AddRange(new ToolStripItem[] { this.toolStripButtonNew });


    // Set the new button
    this.toolStripButtonNew.Image = Bitmap.FromFile("MyNewButton.bmp");
    this.toolStripButtonNew.Size = new System.Drawing.Size(22, 22);
    this.toolStripButtonNew.ToolTipText = "This is my new tool";
    this.toolStripButtonNew.Click +=
        new System.EventHandler(toolStripButtonNew_Click);


    gaiaToolsStripContainer.TopToolStripPanel.Controls.Add(toolStripNew);
}
```

CarbonTools PRO Based Calls

Getting Gaia's CarbonTools based Controls



SetLayersContainer		
Parameters	<i>CarbonTools.Windows.Forms.LayersContainer layersContainer</i> The layers container control used by Gaia	
Expected return types	none	
Sets Gaia's main LayersContainer control that contains all map layers.		
Note: requires CarbonTools PRO developer license		
<pre>public void SetLayersContainer(LayersContainer gaiaLayersContainer) { this.gaiaLayersContainer = gaiaLayersContainer; }</pre>		

SetMultiMap		
Parameters	<i>CarbonTools.Windows.Forms.MultiMap map</i> The main map control	
Expected return types	none	
Sets Gaia's main MultiMap control that contains all map layers.		
Note: requires CarbonTools PRO developer license		
<pre>public void SetMultiMap(MultiMap gaiaMultiMap) { this.gaiaMultiMap = gaiaMultiMap; }</pre>		

SetNotesContainer	
Parameters	<i>CarbonTools.Windows.Forms.NotesContainer notesContainer</i> The notes container control used by Gaia
Expected return types	none
Sets Gaia's main NotesContainer control that contains all user notes.	
Note: requires CarbonTools PRO developer license	
<pre>public void SetNotesContainer(NotesContainer gaiaNotesContainer) { this.gaiaNotesContainer = gaiaNotesContainer; }</pre>	

SetNotesSymbolSelectorDialog	
Parameters	<i>CarbonTools.Windows.Forms.SymbolSelectorDialog symbolSelector</i> The notes pushpin symbol selector dialog used by Gaia
Expected return types	none
Sets the pushpin symbol selection dialog used by Gaia when adding or editing notes. This dialog is activated via the note editor to select the pushpin style.	
Note: requires CarbonTools PRO developer license	
<pre>SymbolLibrary library; SymbolSelectorDialog gaiaNotesSymbolSelector; // Set a new symbol library to the notes pushpin Symbol Selector public void SetNotesSymbolSelectorDialog(SymbolSelectorDialog symbolSelector) { symbolSelector.SymbolLibrary = this.library; this.gaiaNotesSymbolSelector = symbolSelector; }</pre>	

SetSymbolSelectorDialog



Parameters

CarbonTools.Windows.Forms.SymbolSelectorDialog *symbolSelector*
The features symbol selector dialog used by Gaia

Expected return types

none

Sets the features symbol selection dialog used by Gaia. This dialog is used in the feature layer properties as well as when double clicking on the legend or a layer's thumbnail.

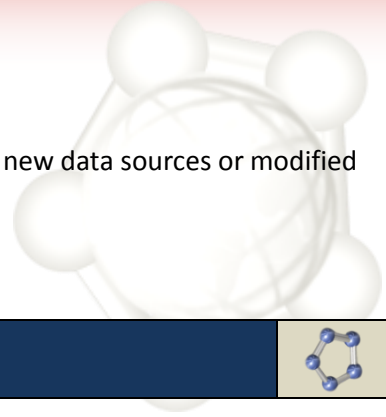
Note: requires CarbonTools PRO developer license


```
SymbolLibrary library;
SymbolSelectorDialog gaiaSymbolSelector;

// Set a new symbol library to the features Symbol Selector
public void SetSymbolSelectorDialog(SymbolSelectorDialog symbolSelector)
{
    symbolSelector.SymbolLibrary = this.library;
    this.gaiaSymbolSelector = symbolSelector;
}
```


Modifying the Layer Info Tool

These methods allow modification of the “Layer Info” dialog, allowing for new data sources or modified source behavior to affect the information view.



GetFeaturesInfoPanel		
Parameters	<p><i>CarbonTools.Data.GeoObject layer</i> The layer selected by the user in the Info dialog</p> <p><i>double worldX, worldY</i> The world coordinates of the info event as selected by the user.</p> <p><i>CarbonTools.Content.Features.ItemList features</i> The features in the <i>layer</i> that match the user selection</p>	
Expected return types	<i>System.Windows.Forms.Control</i>	
<p>This method allows the extender to override the displayed information regarding a features based layer when using the “Info” tool. The returned Control will be displayed in the Info dialog. The extender should inspect the layer and decide whether it is appropriate to override the default Gaia display and return the custom display; otherwise the method should return null and allow Gaia to use the internal display.</p> <p>Gaia will populate the <i>features</i> list with items that match the user’s selection from the selected <i>layer</i>. This saves the user the effort of inspecting manually the complete features set.</p> <p>Note: requires CarbonTools PRO developer license</p>		
<pre>public Control GetFeaturesInfoPanel(GeoObject go, double worldX, double worldY, IList features) { if (features.Count == 0 !(features[0] is MyFeatureClass)) return null; return new MyFeaturesLayerInfoPanel(features); }</pre>		

GetLayerInfoPanel



Parameters	<i>CarbonTools.Data.GeoObject layer</i> The layer selected by the user in the Info dialog <i>double worldX, worldY</i> The world coordinates of the info event as selected by the user.
Expected return types	<i>System.Windows.Forms.Control</i>

This method allows the extender to override the displayed information regarding a layer when using the “Info” tool. The returned Control will be displayed in the Info dialog. The extender should inspect the layer and decide whether it is appropriate to override the default Gaia display and return the custom display; otherwise the method should return **null** and allow Gaia to use the internal display.


Note: requires CarbonTools PRO developer license


```
public Control GetLayerInfoPanel(GeoObject go, double worldX, double worldY)
{
    if (!(go.Handler is HandlerWMS)) return null;

    return new MyWMSInfoPanel(go);
}
```

Modifying the Layer Properties View

These methods allow the modification of the layers “Properties” panel, allowing for new data sources or modified source behavior to affect the properties tabular view.

AcceptLayerProperty		
Parameters	<i>CarbonTools.Data.GeoObject layer</i> The layer displayed in the properties.	
Expected return types	<i>Bool</i> True if the Extender overrides the default tabs for the layer; false if the default tabs should be used in conjunction with the new functionality.	
<p>This method allows performing operations prior to Gaia setting the property tabs for the layer. This method is called before setting the default tabs. In case the method returns true the default property tabs will not be displayed.</p> <p>Note: requires CarbonTools PRO developer license</p> <pre>public void AcceptLayerProperty (GeoObject go) { SourceWFS source = _handler.Source as SourceWFS; source.FilterString = ""; }</pre>		

AcceptLayerPropertyFinal		
Parameters	<i>CarbonTools.Data.GeoObject layer</i> The layer displayed in the properties.	
Expected return types	none	
<p>This method allows performing operations after Gaia sets the property tabs for the layer.</p> <p>Note: requires CarbonTools PRO developer license</p> <pre>public void AcceptLayerPropertyFinal(GeoObject go) { SourceWFS source = _handler.Source as SourceWFS; if (String.IsNullOrEmpty(source.Namespaces) source.Namespaces.IndexOf("xmlns:fes=") < 0) source.Namespaces += " xmlns:fes=\"http://www.opengis.net/fes/2.0\""; }</pre>		

GetLayerPropertyCustomPages



Parameters

CarbonTools.Data.GeoObject layer
The layer displayed in the properties.

Expected return types

TabPage[]
An array of tap pages to add. Return **null** if no pages should be added.

This method allows the Extender to add customized tab pages to be added to the layer's "Properties" page.

Note: requires CarbonTools PRO developer license

```
public TabPage[] GetLayerPropertyCustomPages(GeoObject go)
{
    if (go.Handler is HandlerWFS)
    {
        PropertyPageTemporalFiltering page = new PropertyPageTemporalFiltering(go);

        return new TabPage[] { page as TabPage };
    }
    else
        return null;
}
```

GetLayerPropertyPanelNames



Parameters

CarbonTools.Data.GeoObject layer
The layer displayed in the properties.

Expected return types

string[]
An array of tap pages to add. Return **null** if no pages should be added.

This method allows fine-tuning which of the default tabs should be used in the "Properties" page. The Extender should return a list of tab names from: "query", "extents", "filters", "style", "symbols", "labels", "temporal", and "security". Only the selected tabs will appear.

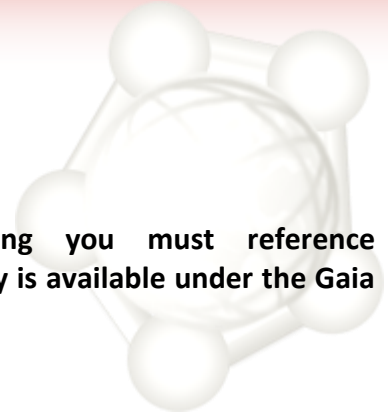
Note: requires CarbonTools PRO developer license

```
public string[] GetLayerPropertyPanelNames(GeoObject go)
{
    if (go.Handler is HandlerWFS New)
        return new string[] { "query", "extents", "filters", "symbols", "labels",
                               "temporal", "security" };
    else
        return null;
}
```

Events



Note: In order to use Gaia's events handling you must reference **Gaia.ExtenderHelper.dll** in your project. This assembly is available under the Gaia installed folder.



Event Handlers

GaiaToExtenderEventHandler

An event handler created by Gaia and used to report events relating to Gaia's functionality. This property is used to get a reference by extenders to the events handler initiated and managed by Gaia. Once attached by the extender callbacks can be handled by the extender.

Supported events:

SetMouseLocation: Get the coordinates according to a mouse-move event.

SetMapBBBox: Get the viewed bounding-box after the map's region changes (e.g. due to zoom or pan)

SessionLoaded: Indicates that a new session files was initiated

ExtenderToGaiaEventHandler

An event handler created by Gaia and used to hook events triggered by the extender. This property is used to get a reference by extenders to the events handler initiated and managed by Gaia. Once attached by the extender callbacks within Gaia can be triggered by the extender.

Supported events:

OnAddFile : Adds a spatial layer from a file.

OnAddGeoObject : Adds a spatial layer from a CarbonTools GeoObject class.

OnChangeCenterMap : Center the map on the given location.

OnChangeMapBBBox : Set the map to the given region.

OnLoadInsertSession : Insert a new GSF to current session.

OnLoadSession : Load a new GSF.

OnMarkSessionAsUnsaved : Instruct Gaia to mark the current session as unsaved.

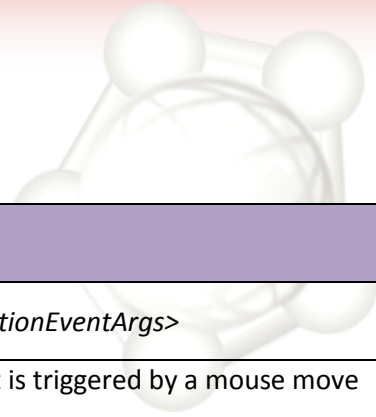
OnRefreshAllLayers : Refresh all the layers in the session.

OnRefreshLayer : Refresh a specific layer.

OnRemoveGeoObject : Remove a layer from session according to its CarbonTools GeoObject.

OnStartNewSession – Instruct Gaia to start a fresh session.

Gaia Map Events



SetMouseLocation

Event Type

EventHandler<Gaia.ExtenderHelper.PositionEventArgs>

Update the extension with a location of the cursor on the map. This event is triggered by a mouse move over the map control.

```
private GaiaToExtenderEvents gaiaToExtenderEventsHandler;
public GaiaToExtenderEvents GaiaToExtenderEventsHandler
{
    set
    {
        this.gaiaToExtenderEventsHandler = value;
        this.gaiaToExtenderEventsHandler.SetMouseLocation +=
            new EventHandler<PositionEventArgs>(SetMouseLocation);
    }
}

void SetMouseLocation(object sender, PositionEventArgs e)
{ // Do something where e.X, e.Y are the world coordinates of the mouse
}
```

SetMapBBox

Event Type

EventHandler<Gaia.ExtenderHelper.BBoxEventArgs>

Update the bounding box region of the map. This method is activated every time the map region changes, usually due to a pan or zoom event.

```
private GaiaToExtenderEvents gaiaToExtenderEventsHandler;
public GaiaToExtenderEvents GaiaToExtenderEventsHandler
{
    set
    {
        this.gaiaToExtenderEventsHandler = value;
        this.gaiaToExtenderEventsHandler.SetMapBBox +=
            new EventHandler<BBoxEventArgs>(SetMapBBox);
    }
}

void SetMapBBox(object sender, BBoxEventArgs e)
{ // Do something where e.X0, e.Y0, e.X1, e.Y1, e.SRS are the region coordinates
}
```

SessionLoaded

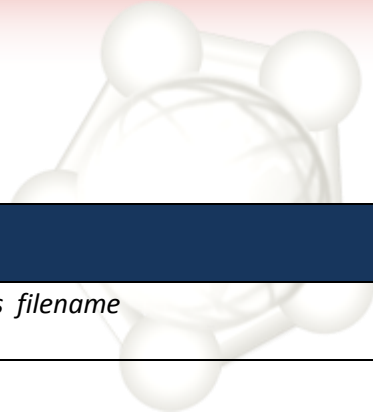
Event Type

EventHandler<Gaia.ExtenderHelper.FileNameEventArgs>

This event is raised when a new Gaia session was loaded.

```
private GaiaToExtenderEvents gaiaToExtenderEventsHandler;  
  
public GaiaToExtenderEvents GaiaToExtenderEventsHandler  
{  
    set  
    {  
        this.gaiaToExtenderEventsHandler = value;  
        this.gaiaToExtenderEventsHandler.SessionLoaded +=  
            new EventHandler<FileNameEventArgs>( SessionLoaded );  
    }  
}  
  
void SessionLoaded(object sender, FileNameEventArgs e)  
{  
    // Do something where e.FileName is the new session's file name  
}
```

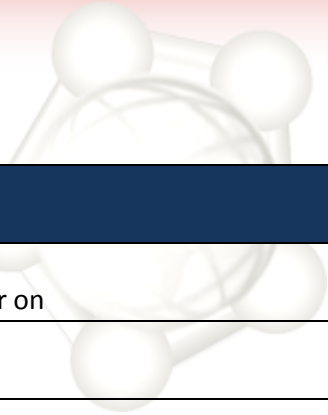
Triggering Gaia Functionality



OnAddFile	
Parameters	<i>Gaia.ExtenderHelper.FileNameEventArgs filename</i> The file name that Gaia should add
Expected return types	none
<p>Instruct Gaia to add a layer based on a file according to its name and title. The title is an optional text used in the layers legend as a human readable title for the layer.</p> <p>Gaia will recognize the file type according to its extension, supported extensions are:</p> <p>GML: gml, gml2, gml3, gml3_2 Shapefiles: shp, shx, dbf MapInfo: mif Autodesk: dxf GoogleEarth: kml, kmz</p>	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void AddLayerFromFile(string filename, double title) { ExtenderToGaiaEventsHandler.OnAddFile(new FileNameEventArgs(filename, title)); }</pre>	

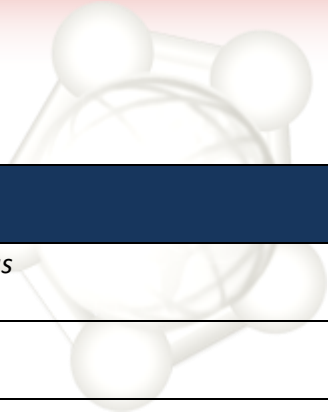
OnAddGeoObject	
Parameters	<i>Gaia.ExtenderHelper.ObjectEventArgs geoObject</i> An object containing a CarbonTools.Data.GeoObject class
Expected return types	none
<p>Instruct Gaia to add a layer using a GeoObject class. A CarbonTools GeoObject class can contain handler, data, renderers and other information about the added layer.</p> <p>Note: requires CarbonTools PRO developer license</p>	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void AddLayer (GeoObject go) { ExtenderToGaiaEventsHandler.OnAddGeoObject(new ObjectEventArgs(go)); }</pre>	





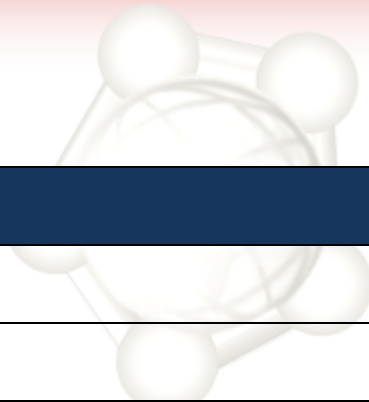
OnChangeCenterMap	
Parameters	<i>Gaia.ExtenderHelper.PositionEventArgs pos</i> The location on which the map should center on
Expected return types	none
Request Gaia to center the map according to a given position.	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void SetMapToLocation(double lat, double lon) { ExtenderToGaiaEventsHandler.OnChangeCenterMap(new PositionEventArgs(lon, lat)); }</pre>	

OnChangeMapBBox	
Parameters	<i>Gaia.ExtenderHelper.BBoxEventArgs bbox</i> The region on which the map should center on
Expected return types	none
Request Gaia to center the map according to a given bounding-box region.	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void SetMapToRegion(double lat0, double lon0, double lat1, double lon1) { ExtenderToGaiaEventsHandler.OnChangeMapBBox(new BBoxEventArgs(lon0, lat0, lon1, lat1, "EPSG:4326")); }</pre>	



OnLoadInsertSession	
Parameters	<i>Gaia.ExtenderHelper.FileNameEventArgs args</i> The session file name to import
Expected return types	none
Request Gaia to import a session file into the current session.	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void ImportGsf() { ExtenderToGaiaEventsHandler.OnLoadInsertSession (new FileNameEventArgs("MySessionNotes.gsf")); }</pre>	

OnLoadSession	
Parameters	<i>Gaia.ExtenderHelper.FileNameEventArgs args</i> The session file name to load
Expected return types	none
Request Gaia to load a session file and discard the current session.	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void LoadGsf() { ExtenderToGaiaEventsHandler.OnLoadSession (new FileNameEventArgs("BaseMaps.gsf")); }</pre>	

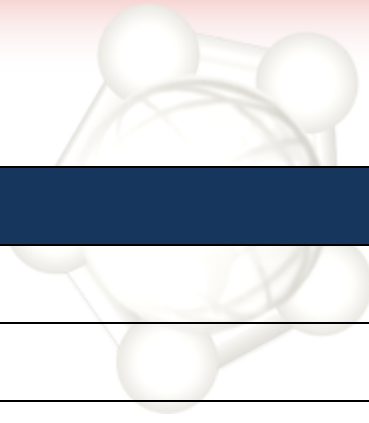


OnMarkSessionAsUnsaved	
Parameters	none
Expected return types	none
Instruct Gaia to mark the current session as “dirty”, thus enabling the “Save” menu item and popping the save session verification when the session is closed.	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void MarkSessionAsUnsaved() { ExtenderToGaiaEventsHandler.OnMarkSessionAsUnsaved(); }</pre>	

OnRefreshAllLayers	
Parameters	none
Expected return types	none
Instruct Gaia to perform the “Refresh All” function.	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void OnRefreshAllLayers() { ExtenderToGaiaEventsHandler.OnRefreshAllLayers(); }</pre>	

OnRefreshLayer	
Parameters	<i>Gaia.ExtenderHelper.ObjectEventArgs geoObject</i> An object containing a CarbonTools.Data.GeoObject class
Expected return types	none
Instruct Gaia to refresh a specific layer in the session.	
Note: requires CarbonTools PRO developer license	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void RefreshLayer(GeoObject go) { ExtenderToGaiaEventsHandler.OnRefreshLayer(new ObjectEventArgs(go)); }</pre>	

OnRemoveGeoObject	
Parameters	<i>Gaia.ExtenderHelper.ObjectEventArgs geoObject</i> An object containing a CarbonTools.Data.GeoObject class
Expected return types	none
Instruct Gaia to remove a layer from the session.	
Note: requires CarbonTools PRO developer license	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void RemoveLayer(GeoObject go) { ExtenderToGaiaEventsHandler.OnRemoveGeoObject(new ObjectEventArgs(go)); }</pre>	



OnStartNewSession	
Parameters	none
Expected return types	none
Instruct Gaia to start a new session.	
<pre>public ExtenderToGaiaEvents ExtenderToGaiaEventsHandler { get; set; } private void StartNew() { ExtenderToGaiaEventsHandler.OnStartNewSession(); }</pre>	