



# Geocortex Viewer for HTML5 2.9

---

Administrator and Developer Guide



© 2017 Latitude Geographics Group Ltd.

All Rights Reserved.

**Printed in Canada**

The information contained in this document is the exclusive property of Latitude Geographics Group Ltd. and/or its licensors. This work is protected under Canadian and US copyright law and copyright laws of the given countries of origin and applicable international laws, treaties, and/or conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage or retrieval system, except as expressly permitted in writing by Latitude Geographics Group Ltd. All requests should be sent to Attention: Contracts Manager, Latitude Geographics Group Ltd, 300 - 1117 Wharf Street, Victoria, British Columbia V8W 1T7.

This information is subject to change without notice.

**US Government Restricted Rights**

*The Software and documentation are provided with restricted rights. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013, or subparagraphs (c)(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. The Owner or authorized licensor is Latitude Geographics Group Ltd., 300 - 1117 Wharf Street Victoria, British Columbia V8W 1T7.*

Geocortex and Latitude Geographics are registered trademarks of Latitude Geographics Group Ltd. in the United States and Canada, and are trademarks in other jurisdictions around the world. Esri, the Esri globe logo, ArcGIS, @esri.com, and esri.com are trademarks, service marks, or registered marks of Esri in the United States, the European Community, or certain other jurisdictions. Trademarks are provided under license from Esri. Microsoft and the Windows logo are registered trademarks of Microsoft Corporation. Other companies and products mentioned herein are trademarks or registered trademarks of their respective trademark owners.

---

# Contents

<b>1</b>	<b>Welcome</b>	<b>1</b>
1.1	About the Geocortex Viewer for HTML5	1
1.2	About this Guide	1
1.3	Disclaimer	2
<b>2</b>	<b>Getting Help</b>	<b>2</b>
2.1	Online Resources	2
2.2	Email Support	2
2.3	Viewer Log	3
2.4	Provide Feedback on Documentation	4
<b>3</b>	<b>Requirements</b>	<b>4</b>
3.1	Client Requirements	4
3.1.1	Desktop Browser Support	5
3.1.2	Mobile Browser Support	5
3.2	Web Server Requirements	6
3.3	Geocortex Essentials Requirements	6
3.4	ArcGIS Requirements	6
3.5	Development Requirements and Suggestions	6
<b>4</b>	<b>Viewer Installation Options</b>	<b>7</b>
4.1	About Installation Using a Viewer Template	7
4.2	About Manual Installation	7
<b>5</b>	<b>Download the Installation Package</b>	<b>8</b>
5.1	Contents of the Installation Package	9
<b>6</b>	<b>Launch the Post Installer</b>	<b>9</b>
<b>7</b>	<b>First-Time Installation</b>	<b>10</b>
7.1	Install the Viewer Framework Using a Viewer Template	10
7.1.1	Add a Viewer to a Site	13
7.2	Install the Viewer Manually	16
7.2.1	Install the Viewer	16
7.2.2	Launch the Viewer	16
7.2.3	Point the Viewer to Your Site	17
<b>8</b>	<b>Upgrading</b>	<b>19</b>
8.1	Upgrade Options	19
8.1.1	About Upgrading Viewers Using Essentials	19
8.1.2	About Upgrading the Viewer Manually	21
8.2	Versions that Require Manual Steps	21

<b>8.3 Upgrade Viewers Using Essentials</b>	<b>24</b>
8.3.1 Step 1: Upgrade the Viewer Template	24
8.3.2 Step 2: Upgrade your Configured Viewers	26
<b>8.4 Restore a Viewer</b>	<b>28</b>
<b>8.5 Upgrade the Viewer Manually</b>	<b>30</b>
<b>9 Set Up a Proxy Page</b>	<b>31</b>
9.1 Use the ASP.NET Proxy Page that Ships with the Viewer	32
9.2 Adapt the ASP.NET Proxy Page to Access Windows-Secured Services	33
<b>10 Set Up Cross-Origin Resource Sharing (CORS)</b>	<b>34</b>
<b>11 Architecture</b>	<b>40</b>
11.1 About HTML5 Architecture	40
11.2 About HTML5 Applications	40
11.3 Multi-Device Performance	40
11.4 Modules, Views, and View Models	40
<b>12 Viewer Launch URLs</b>	<b>41</b>
12.1 About Viewer URLs	41
12.2 Short Viewer URLs	44
12.3 URL Parameters Reference	49
12.4 URLs for Secured Sites	54
<b>13 About Viewer Configuration</b>	<b>57</b>
13.1 Configuration Files	57
13.2 Structure of Configuration Files	58
13.2.1 Application-Wide Settings	58
13.2.2 Libraries and the Default Library ID	60
13.2.3 Modules	61
13.2.4 Views	61
13.2.5 View Models	61
13.3 General Method for Manual Configuration	61
13.4 About Configuration Using Manager	62
13.4.1 About Configuring Multiple Interfaces	62
13.4.2 About the Live Preview	64
13.4.3 About User Interface Text	64
<b>14 Configure a Viewer in Manager</b>	<b>65</b>
14.1 Add a Viewer to a Site	65
14.2 Configure the Splash Screen	65
14.3 Edit a Viewer	67
14.4 Change Viewer Information	68
14.5 Configure Application-Wide Settings	70



---

<b>14.6 Configure Accessibility</b>	<b>72</b>
<b>14.7 Change the Look and Feel</b>	<b>74</b>
<b>14.8 Configure the I Want To Menu</b>	<b>79</b>
<b>14.9 Configure the Context Menus</b>	<b>81</b>
14.9.1 Configure the Map Context Menu	85
<b>14.10 Configure Measurement</b>	<b>86</b>
<b>14.11 Configure the Map</b>	<b>87</b>
<b>14.12 Configure Map Widgets</b>	<b>90</b>
<b>14.13 Configure a Viewer for Offline Use</b>	<b>94</b>
<b>14.14 Configure Geolocation</b>	<b>95</b>
<b>14.15 Configure the Home Panel</b>	<b>99</b>
<b>14.16 Configure Instant Search in the HTML5 Viewer</b>	<b>102</b>
<b>14.17 Configure the Layer List</b>	<b>103</b>
<b>14.18 Configure Pushpins</b>	<b>104</b>
<b>14.19 Configure the Toolbar</b>	<b>108</b>
<b>14.20 Configure Tool Behavior</b>	<b>121</b>
<b>14.21 Configure Optimizer Integration</b>	<b>123</b>
<b>14.22 Configure Analytics Integration</b>	<b>124</b>
<b>14.23 Configure Collaboration</b>	<b>126</b>
14.23.1 Working with Multiple Rooms	128
<b>15 Configuration Settings by Module</b>	<b>129</b>
<b>15.1 Common Settings for Modules</b>	<b>129</b>
<b>15.2 Common Settings for Views</b>	<b>130</b>
<b>15.3 Common Settings for View Models</b>	<b>130</b>
<b>15.4 Accessibility Module</b>	<b>131</b>
<b>15.5 Alert Module</b>	<b>133</b>
<b>15.6 Authentication Module</b>	<b>134</b>
<b>15.7 Banner Module</b>	<b>134</b>
<b>15.8 BarcodeScanner Module</b>	<b>135</b>
<b>15.9 Basemap Module</b>	<b>136</b>
<b>15.10 Bookmarks Module</b>	<b>138</b>
<b>15.11 Browser Module</b>	<b>140</b>
<b>15.12 Buffer Module</b>	<b>140</b>
<b>15.13 Charting Module</b>	<b>143</b>
<b>15.14 ClusterLayers Module</b>	<b>148</b>
<b>15.15 Collaboration Module</b>	<b>149</b>
<b>15.16 CompactToolbar Module</b>	<b>150</b>
<b>15.17 Confirm Module</b>	<b>158</b>
<b>15.18 Coordinates Module</b>	<b>159</b>
<b>15.19 Editing Module</b>	<b>160</b>
<b>15.20 ExportMap Module</b>	<b>165</b>
<b>15.21 ExportWebMap Module</b>	<b>166</b>
<b>15.22 FeatureDetails Module</b>	<b>168</b>

---

15.23	FeatureLayer Module	176
15.24	Footer Module	176
15.25	Geolocate Module	184
15.26	GlobalMenu Module	187
15.27	HeatMaps Module	189
15.28	Highlight Module	191
15.29	Identify Module	193
15.30	Info Module	196
15.31	InsightIntegration Module	198
15.32	Integration Module	200
15.33	IWantToMenu Module	215
15.34	LabelOptions Module	218
15.35	LayerAddition Module	222
15.36	LayerCatalog Module	226
15.37	LayerDrawingOrder Module	228
15.38	LayerList Module	232
15.39	LayerStyles Module	235
15.40	LayerThemes Module	242
15.41	Legend Module	244
15.42	Log Module	245
15.43	Map Module	246
15.44	MapTips Module	253
15.45	Markup Module	258
15.46	Measurement Module	269
15.47	Menu Module	282
15.48	NativeIntegration Module	284
15.49	Navigation Module	285
15.50	Offline Module	287
15.51	Offline Maps Module	288
15.52	OptimizerIntegration Module	294
15.53	OverviewMap Module	295
15.54	PlotCoordinates Module	297
15.55	Printing Module	302
15.56	Project Module	304
15.57	Prompt Module	316
15.58	Pushpins Module	317
15.59	QueryBuilder Module	320
15.60	Reporting Module	327
15.61	Results Module	328
15.62	Scalebar Module	337
15.63	Search Module	339
15.64	Selection Module	341
15.65	Share Module	345
15.66	SharingLink Module	347

---

15.67 Shells Module	350
15.68 SimpleLayerList Module	358
15.69 Site Module	359
15.70 SkipLinks Module	359
15.71 Snapping Module	360
15.72 Status Module	363
15.73 TabbedToolbar Module	363
15.74 TimeSlider Module	372
15.75 Tools Module	375
15.76 UploadData Module	377
15.77 User Module	389
15.78 Visualization Module	390
15.79 Workflow Module	392
15.80 WorkflowHost Module	394
15.81 ZoomControl Module	395
<b>16 Viewer Commands</b>	<b>396</b>
<b>16.1 About Commands</b>	<b>396</b>
16.1.1 Viewer Commands in Hyperlinks	396
16.1.2 Viewer Commands in Workflows	397
<b>17 Editing</b>	<b>397</b>
<b>18 Offline</b>	<b>401</b>
<b>18.1 Configure Offline Support</b>	<b>402</b>
<b>19 Accessibility</b>	<b>402</b>
<b>19.1 Screen Readers</b>	<b>403</b>
<b>19.2 Keyboard Shortcuts</b>	<b>403</b>
<b>20 Protecting Against Malicious Code</b>	<b>407</b>
<b>21 Translation</b>	<b>410</b>
<b>21.1 About Translating UI Text</b>	<b>410</b>
<b>21.2 Translate Language Files</b>	<b>410</b>
<b>22 Custom Development</b>	<b>413</b>
<b>22.1 About Custom Development</b>	<b>413</b>
<b>22.2 Key Concepts</b>	<b>414</b>
22.2.1 HTML and JavaScript	414
22.2.2 Separation of Concerns	414
22.2.3 Model-View-ViewModel (MVVM)	415
22.2.4 Data Binding in HTML	416
22.2.5 Libraries and Resource Compilation	417
22.2.6 UI Composition	418

---

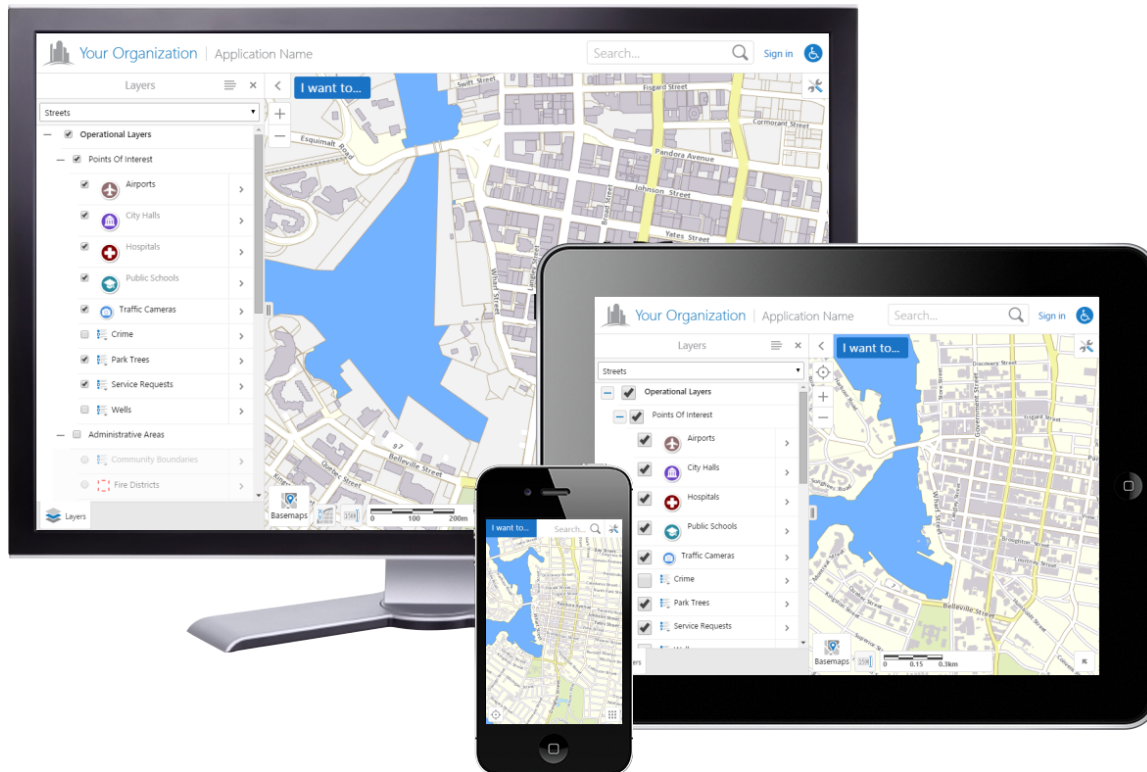
22.2.7	Commands and Events	420
22.2.8	States	420
<b>22.3</b>	<b>Samples Viewer</b>	<b>421</b>
<b>22.4</b>	<b>Geocortex SDK for HTML5 API Reference</b>	<b>422</b>
<b>22.5</b>	<b>SDK and QuickStart</b>	<b>423</b>
22.5.1	Basic Viewer	423
<b>22.6</b>	<b>Project Layout</b>	<b>423</b>
<b>22.7</b>	<b>Build</b>	<b>424</b>
<b>22.8</b>	<b>Application Life Cycle</b>	<b>424</b>
<b>22.9</b>	<b>Some Notes on JavaScript</b>	<b>425</b>
<b>22.10</b>	<b>The Resource Compiler</b>	<b>425</b>
22.10.1	Resource Compiler Tool	425
22.10.2	Resource Manifests	426
<b>22.11</b>	<b>Data Binding</b>	<b>426</b>
22.11.1	About Data Binding	426
22.11.2	Basics	426
22.11.3	The Observable Type	427
22.11.4	Attribute Binding	428
22.11.5	Text Binding	429
22.11.6	Visibility Bindings	429
22.11.7	Events	429
22.11.8	ObservableCollection	430
22.11.9	Collections	430
<b>22.12</b>	<b>HTML5 Viewer Embedding</b>	<b>432</b>
22.12.1	About HTML5 Viewer Embedding	432
22.12.2	Embed an HTML5 Viewer within an Iframe	435
22.12.3	Embed an HTML5 Viewer within the Entire Page	435
<b>Appendix A:</b>	<b>Glossary</b>	<b>441</b>
<b>Appendix B:</b>	<b>Regions</b>	<b>444</b>
<b>B.1</b>	<b>Regions in the Desktop and Tablet Interfaces</b>	<b>444</b>
<b>B.2</b>	<b>Regions in the Handheld Interface</b>	<b>445</b>
<b>Appendix C:</b>	<b>File Locations</b>	<b>448</b>
<b>Appendix D:</b>	<b>Command Reference</b>	<b>450</b>
<b>Appendix E:</b>	<b>Event Reference</b>	<b>451</b>
<b>Appendix F:</b>	<b>State Reference</b>	<b>452</b>
<b>F.1</b>	<b>Global States</b>	<b>452</b>
<b>F.2</b>	<b>Non-global States</b>	<b>452</b>

# 1 Welcome

Welcome to the Geocortex Viewer for HTML5 2.9 Administrator and Developer Guide.

## 1.1 About the Geocortex Viewer for HTML5

The Geocortex Viewer for HTML5 is a web mapping application framework geared towards creating clean and effective mobile web applications for a wide variety of browsers and devices. The HTML5 Viewer has been designed from the ground up to make it possible to use the same code base across many different platforms and device types without investing in extensive platform-specific development. Instead, much of a viewer's appearance and interaction across different platforms are handled by configuration files and Cascading Style Sheet (CSS) files. The Geocortex Viewer for HTML5 allows administrators and developers to deploy robust, targeted web applications across a wide variety of platforms with ease.



The HTML5 Viewer on multiple devices

## 1.2 About this Guide

This Administrator and Developer Guide explains how to configure and develop viewers using the Geocortex Viewer for HTML5 framework.

## 1.3 Disclaimer

The Geocortex HTML5 SDK and the Geocortex Viewer for HTML5 work with other licensed software products. You are responsible for ensuring that all of the required software licenses are in place and appropriate for your chosen system configuration.

Refer to your Geocortex Viewer for HTML5 software license agreement for information about limitations on liability and other legal considerations.

## 2 Getting Help

### 2.1 Online Resources

- [Geocortex Viewer for HTML5 Community](#) on the Geocortex Support Center
- Esri documentation for the [ArcGIS API for JavaScript](#)

### 2.2 Email Support

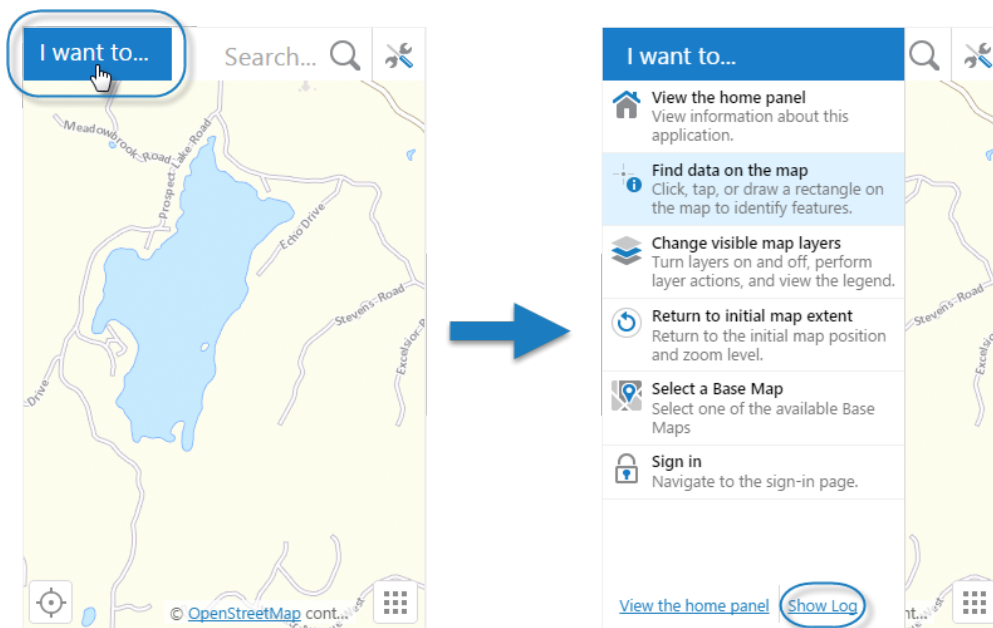
If you have a question that is not suitable for posting to the community, and you have an active Support agreement, send an email message to the Geocortex Support team at [support@geocortex.com](mailto:support@geocortex.com). The Support team will respond by email.

## 2.3 Viewer Log

The Viewer keeps a log of its activities. This log is helpful when tracking down incompatibilities and errors. Including relevant log entries in a Geocortex community post or email to Geocortex Support can be very useful for troubleshooting. The viewer log is only available while the browser window remains open.

### ► To view the log:

- **On a desktop PC or tablet:** Do one of the following:
  - Long-press the banner at the top.
  - Press **Ctrl + Shift + ~**.
  - Hold down the **Shift** key and press **Esc** twice.
- **On a handheld device:** Tap the **I Want To** menu button, and tap **Show Log**.



The I Want To menu button in the Handheld interface (left), and the Show Log link

### ► To filter the log:

1. [View the log](#).
2. Click or tap **Filter View**.  
Several filters appear.

3. Click or tap one of the following filters:
  - **Errors:** Display only errors. An error is the most serious type of log entry.
  - **Warnings:** Display only warnings, and more serious log entries.
  - **Info:** Display all log entries, except debug entries. This is the default filter.
  - **Debug:** Display all log entries, including debug entries. This filter can only be used if you [enable debug logging](#).

▶ **To enable debug logging:**

Debug logging provides extra information about the viewer's activities. While this can be useful for troubleshooting a problem, it can cause reduce viewer performance. We do not recommend enabling debug logging for normal use of the viewer.

1. [View the log](#).
2. Select the **Enable Debug Logging** checkbox. By default, this checkbox is cleared.  
To actually view debug entries, you must click or tap **Filter View**, and then **Debug**.

▶ **To clear the log:**

1. [View the log](#).
2. Click or tap **Clear Log**.  
The log is cleared for the current session.

## 2.4 Provide Feedback on Documentation

Please send comments and suggestions related to documentation to: [documentation@geocortex.com](mailto:documentation@geocortex.com).

# 3 Requirements

## 3.1 Client Requirements

The Geocortex Viewer for HTML5 is designed to run on many platforms, with support for both recent and older browsers, and with a focus on lightweight mobile development.

To run a Geocortex HTML5 viewer, the client's browser must:

- Support JavaScript.
- Be supported by the Esri ArcGIS API for JavaScript.



The HTML5 Viewer's screen-reading [accessibility](#) features require the Freedom Scientific [JAWS screen reader](#). Other screen reader software may also work but are not officially supported.





The File Attachments feature requires a browser that supports HTML5 File Reader. For a list of browsers that support HTML5 File Reader, see [caniuse.com/#feat=filereader](https://caniuse.com/#feat=filereader).



Browser support for HTML5 varies considerably. Firefox and Chrome tend to offer the broadest feature support and the best performance.

### 3.1.1 Desktop Browser Support

The Geocortex Viewer for HTML5 is tested and can be used on:

- Google Chrome (current version recommended)
- Mozilla Firefox (current version recommended)
- Microsoft Edge (current version preferred)
- Microsoft Internet Explorer 9.0+ (11.0 preferred)



**Starting in version 2.6, the HTML5 Viewer does not support Internet Explorer 8.**



The HTML5 Viewer does not support Internet Explorer's Enterprise mode—Enterprise mode emulates Internet Explorer 8, which is not supported.



Firefox and Chrome do not support SSL 3.0. Internet Explorer 11 has SSL 3.0 turned off by default. If you use SSL encryption to secure Geocortex web applications, we recommend that you use a version that is newer than SSL 3.0.

### 3.1.2 Mobile Browser Support

The Geocortex Viewer for HTML5 is tested and recommended for use on:

- Safari on iOS 9+



**Effective with Geocortex Viewer for HTML5 version 2.9, iOS 9 or greater is required. iOS 10 is recommended.**

- Chrome on Android

## 3.2 Web Server Requirements

The Geocortex Viewer for HTML5 works with any web server capable of serving static content over HTTP.



You need to set up a proxy page for your HTML5 viewers to use. The proxy page enables the viewer to run workflows and perform editing. As well, proxies support large requests. See [Set Up a Proxy Page on page 31](#) for information.

## 3.3 Geocortex Essentials Requirements

Geocortex Viewer for HTML5 2.9 works with Geocortex Essentials 4.8 and later.



If you intend to use the Geocortex Mobile App Framework with the Geocortex Viewer for HTML5 2.9, use the Geocortex Mobile App Framework 2.3.

## 3.4 ArcGIS Requirements

The Geocortex Viewer for HTML5 works with:

- ArcGIS Server 10.1
- ArcGIS Server 10.2
- ArcGIS Server 10.2.1
- ArcGIS Server 10.2.2
- ArcGIS Server 10.3
- ArcGIS Server 10.3.1
- ArcGIS Server 10.4
- ArcGIS Server 10.5

The Geocortex Viewer for HTML5 2.9 uses the ArcGIS API for JavaScript 3.20.



Using a different version of the ArcGIS JavaScript API can lead to unpredictable results.

## 3.5 Development Requirements and Suggestions

The resource compiler included in the SDK (Software Development Kit) requires Java 5 and later. The Java home path must be in the PATH environment variable.

Other than that, a text editor and a web browser is all that is needed.



We recommend browsers that provide robust debugging and inspection tools, such as Chrome or Firefox. An HTTP debugger that allows you to observe and analyze HTTP traffic to and from applications is also useful.



We recommend and use TypeScript 2.1.

## 4 Viewer Installation Options

There are two ways to install the HTML5 Viewer framework:

- **Use the Viewer Template:** Use the Geocortex Essentials Post Installer to add the viewer template (.vte file).
- **Install the Viewer Manually:** Drop the viewer directly into a web server.

The method that you choose has important implications for how you manage your viewers. Read about the two options below before making a decision. Then choose the method that best suits your needs and follow the instructions for that method:

- [Install the Viewer Framework Using a Viewer Template on page 10](#), or
- [Install the Viewer Manually on page 16](#)

### 4.1 About Installation Using a Viewer Template

This method allows you to:

- **Install the Viewer Framework:** This method uses the Post Installer to deploy the viewer framework to Internet Information Services (IIS). The Post Installer performs the configuration in IIS—you do not need to do any work in IIS yourself.
- **Install the Management Pack:** The management pack integrates the viewer framework with Manager. This allows you to use Manager to add viewers to sites and configure viewer properties.

This method also allows you to use Essentials to upgrade your viewers. Using Essentials, you can upgrade all your HTML5 viewers in all your sites with one click.

### 4.2 About Manual Installation



This method is for expert users with a good understanding of web servers and a willingness to manage their viewers by hand.

When you install a viewer manually, the viewer is not integrated with Manager. This means that it does not appear in your site configuration. To configure the viewer, you must manually edit the viewer's configuration files. Manual configuration is prone to error. If you introduce an error into a configuration file, the file may be unusable.

You cannot use Essentials to upgrade manually installed viewers.

The viewer can be deployed using any web server. With most web servers, simply place the contents of the installation package into a folder that is known to the web server. This is enough to get the viewer application up and running.

Manual installation allows you greater control than installation using the viewer template. It allows you to:

- Host your viewers on a different server than Essentials.
- Use a platform and web server other than Windows and IIS to host your viewers. For example, you could use Apache on Linux, or any other platform and web server that you choose.
- Control the URL that users use to open the viewer.

## 5 Download the Installation Package

If you have not already done so, read [Viewer Installation Options on page 7](#) and decide which installation method you want to use.



If you have decided to install the HTML5 Viewer using a viewer template, download the installation package to the server where Essentials is installed—the Viewer and Essentials must be installed on the same server if you want to use Manager to configure your viewers. You must install Essentials before the Viewer.

The installation package for the HTML5 Viewer is available on the [Geocortex Support Center](#).

### ► To download the installation package for the HTML5 Viewer:

1. Open the [Geocortex Essentials Downloads](#) page in the Geocortex Support Center.



If prompted, log in with your Support Center account credentials. If you do not have a Support Center account, click the link to create an account.

2. Find the **Geocortex Viewer for HTML5** section.

Information about the current version of the viewer is displayed, with links to documentation and a button to download the viewer installation package.



To download a previous version, click **Previous Versions** and navigate to the version you want.

3. Click **Download**, and save the file to the desired folder.
4. Extract the files from the installation package.  
Familiarize yourself with the installation package by reading [Contents of the Installation Package on page 9](#).

## 5.1 Contents of the Installation Package

The Geocortex Viewer for HTML5 installation package is distributed as a ZIP file that contains all the files necessary to deploy the viewer framework and supporting resources. The ZIP file contains the following:

- **Software:**
  - **Geocortex.Essentials.HTML5Viewer.Template.2.9.vte:** The Viewer Template Engine file, or "viewer template" for short. The viewer template enables the HTML5 Viewer to integrate with Geocortex Essentials Manager so you can create and configure HTML5 viewers in Manager.
  - **Viewer.zip:** The Geocortex Viewer for HTML5 application and sample proxy pages.
- **SDK:**
  - **QuickStart:** A complete viewer application package designed to help developers get started with custom development.
  - **Framework:** The core framework used to develop with the Geocortex Viewer for HTML5.
  - **Samples:** Ready-to-run samples of the HTML5 Viewer that illustrate concepts and practices to use when doing custom development. Also includes the HTML5 Viewer's Developer Reference, in the Samples Viewer.
  - **Tools:** Development tools such as a resource compiler.
- **Documentation:**
  - **Administrator and Developer Guide.pdf:** A printable document on how to install, configure, and develop with the Geocortex Viewer for HTML5.
  - **Geocortex SDK for HTML5 API Reference:** This documentation is available at <http://gedemo.geocortex.com/SamplesViewer/>.
  - **Installation Guide.pdf:** A printable document on how to install the Geocortex Viewer for HTML5.
  - **README.txt:** A short list of the contents of the ZIP file and information about this release of the Viewer.
  - **Release Notes.pdf:** Information about changes included in the most recent release and past releases of the Geocortex Viewer for HTML5.

## 6 Launch the Post Installer

The Post Installer is used for installing and upgrading viewer templates.

 **To launch the Post Installer:**

Follow the instructions for the operating system you are using:

- **Windows Server 2012 or Windows 8, and Newer Versions:**

On the **Start** screen, type **Post Install**, and then click **Post Installer**.

- **Windows Server 2008 or Windows 7, and Older Versions:**

In the **Start** menu, click **All Programs | Latitude Geographics | Geocortex Essentials [Version] [Instance] | Post Installer**.

[Version] is the Essentials version number. [Instance] is the instance name, if Essentials is installed as a named instance. The default installation does not have an instance name.

The Post Installer opens.

## 7 First-Time Installation

### 7.1 Install the Viewer Framework Using a Viewer Template

These instructions describe the steps to install the Geocortex Viewer for HTML5 using a viewer template.

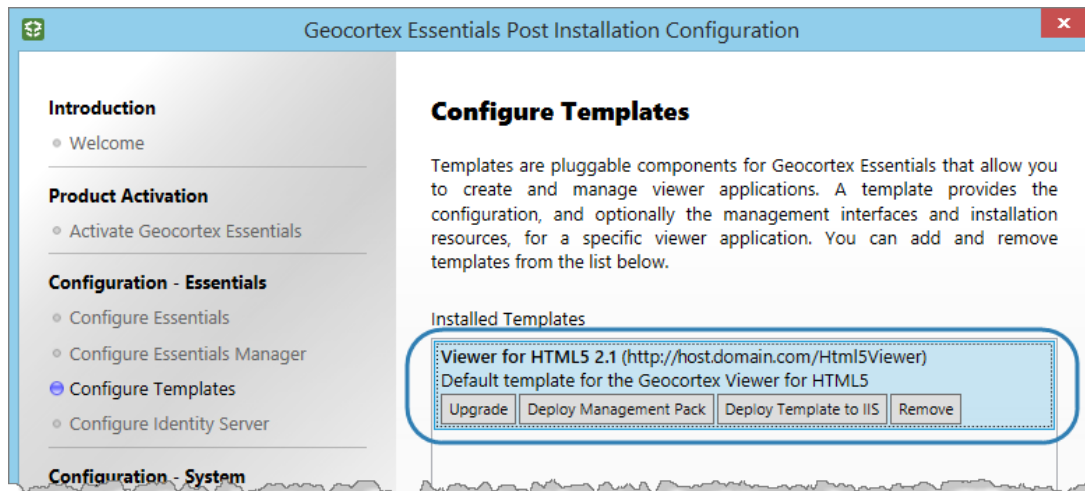


If you are installing the HTML5 Viewer as part of a new installation of Geocortex Essentials, you must install Essentials first.

Installation is done using the Essentials Post-Installation Configuration tool. There are three main steps:

- **Add the template:** This adds the HTML5 Viewer template to the Post-Installation Configuration tool.
- **Install the Management Pack:** This copies the Management Pack files to Manager so you can add and configure HTML5 viewers in Manager.
- **Deploy the viewer template to IIS:** This creates a virtual directory in IIS and copies the HTML5 Viewer's files to the virtual directory so you can launch the viewer in a browser.

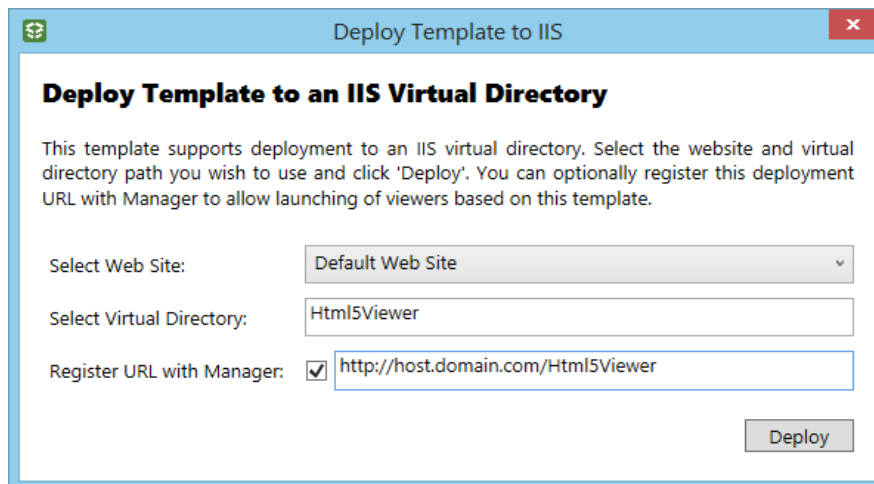
You can add the template to the Post-Installation Configuration tool without installing the Management Pack or deploying the viewer template to IIS. You can install the Management Pack later using the Post-Installation Configuration tool's **Deploy Management Pack** button. Similarly, you can deploy the viewer template to IIS later using the **Deploy Template to IIS** button.



After adding the template, click the template to show the buttons

#### ▶ To install the Geocortex Viewer for HTML5 using Geocortex Essentials:

1. Launch the Post Installer.
  - **Windows Server 2012 or Windows 8, and Newer Versions:**  
On the **Start** screen, type **Post Install**, and then click **Post Installer**.
  - **Windows Server 2008 or Windows 7, and Older Versions:**  
In the **Start** menu, click **All Programs | Latitude Geographics | Geocortex Essentials [Version] [Instance] | Post Installer**.  
[Version] is the Essentials version number. [Instance] is the instance name, if Essentials is installed as a named instance. The default installation does not have an instance name.
2. Click **Configure Templates** in the side panel.
3. Click **Add**.
4. Browse to the folder where you extracted the files from the installation package.
5. Select the template file (.vte file), and then click **Open**.  
You are prompted to install the Management Pack.
6. Click **Yes**.  
You are prompted to deploy the template to an IIS virtual directory.
7. Click **Yes**.  
The Deploy Template to IIS dialog box opens.



Deploy Template to IIS dialog box

8. If you want to deploy the viewer to a website that does not exist yet, create the website in IIS.
9. In the **Select Web Site** box, select the website that you want to deploy the viewer to.
10. In the **Select Virtual Directory** box, type a name or path for the virtual directory where you want the viewer to be deployed.

The URL in the box below the Select Virtual Directory box updates as you type the virtual directory.



The virtual directory must not already exist. Essentials will create the web folder needed to deploy the viewer.



The URL you register with Essentials must have the same domain as Essentials—the viewer cannot launch or load configuration from a different domain.

11. If you want to be able to launch the viewers that are based on this template from Manager, make sure the **Register URL with Manager** checkbox is selected.  
If you clear the **Register URL with Manager** checkbox, the viewer launch links do not appear in Essentials. You will still be able to use Essentials to add and configure viewers.
12. If the URL beside the checkbox is not fully qualified, replace it with the fully qualified URL.  
A fully qualified URL specifies the host and domain, for example, `host.domain.com`. A URL that omits the domain is not fully qualified.  
If you do not specify the fully qualified URL here, the viewer will initially be blank when you launch it from Manager—you will have to manually enter the fully qualified URL in your browser's address bar to see the map. Also, Manager's Preview links will not work.



Geocortex Essentials 3.11.1 and later versions provide the fully qualified URL by default.

13. Click **Deploy**, and then close the success message that displays.  
The template is listed in the Installed Templates area.



- Click **Finish**. If prompted to review your settings, click **OK**, and then click **OK** again to close the Post Installer.

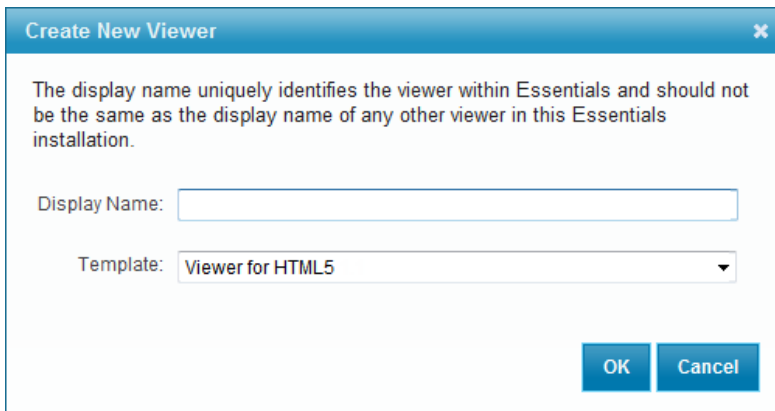
## 7.1.1 Add a Viewer to a Site

### ▶ To add an instance of the HTML5 Viewer to an Essentials site:

- Click the **Windows Start** button, type **Essentials**, and click **Geocortex Essentials Manager**.
- If you are prompted to sign in, enter your user name and password, and then click **Sign In**.  
Depending on how Essentials is configured in the Post Installer, you either sign in using an ArcGIS account or a Windows account.  
Manager opens to the Site List.


 If no sites exist, add a new site.


- Edit the site that you want to add a viewer to.
- In the sidebar, click **Viewers**.  
The Viewers page opens.
- Click **Add Viewer**.  
The Create New Viewer dialog box opens.



Create New Viewer dialog box

- In the **Display Name** box, type a name for the viewer.

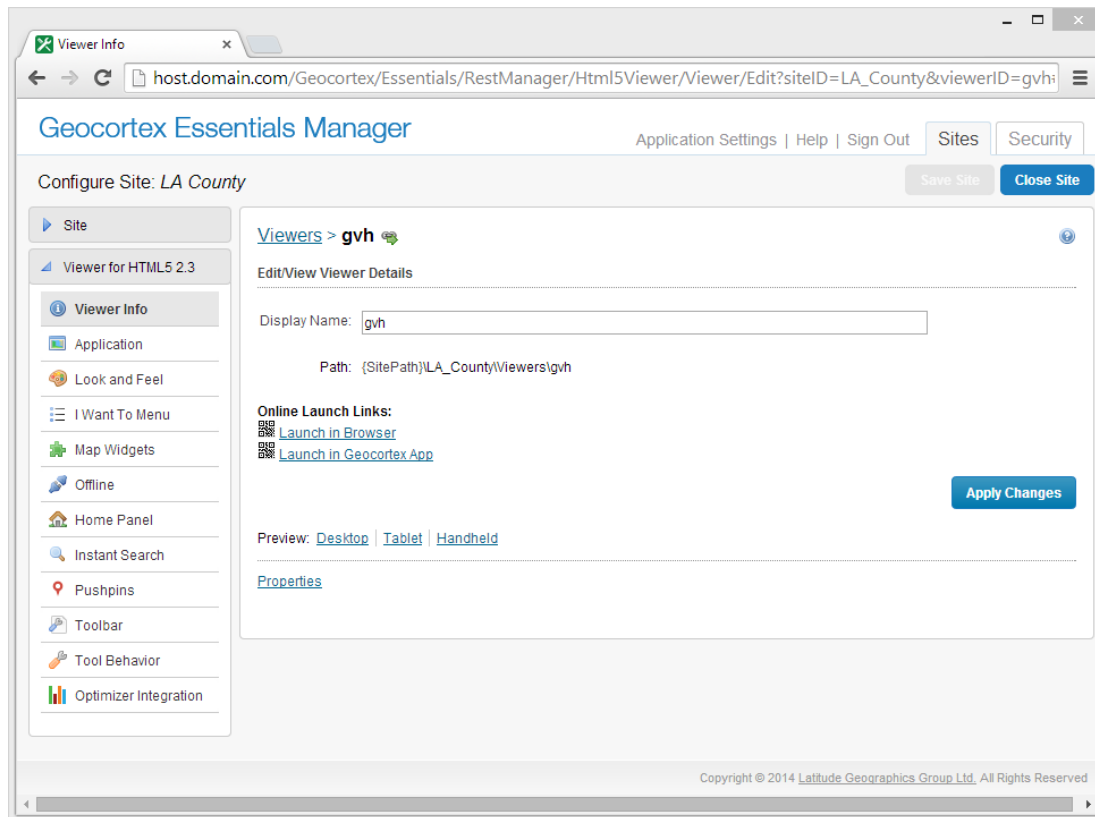
 Avoid long viewer display names. Display names affect Windows file paths, which must be less than 260 characters.

 We recommend using a unique display name for the viewer to prevent confusion.

- In the **Template** box, select **Viewer for HTML5 2.9** from the drop-down list.

8. Click **OK**.


The Create New Viewer dialog box closes and Manager's Viewer Info page opens.



#### Manager's Viewer Info page showing the new viewer

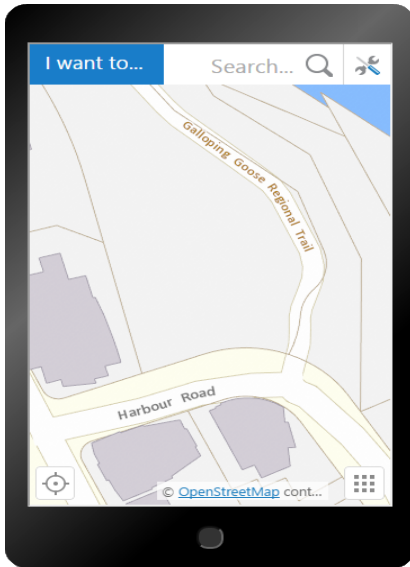
9. To verify your installation, launch the viewer from Manager.

Manager provides several methods of launching the viewer, including:

- Click one of the launch links on the Viewer Info page.
- Hover the pointer over one of the QR codes (  ) to enlarge the code, and then scan it with a device, such as a smartphone.



If the viewer does not load in the browser window, edit the URL in the address bar to fully specify the server and domain, for example, **myserver.companydomain.com**. If the URL uses just the server name, the viewer will not display in some cases.



HTML5 Viewer's Handheld interface

## 7.2 Install the Viewer Manually

These instructions describe the steps to manually install the Geocortex Viewer for HTML5.

### 7.2.1 Install the Viewer



In the method described here, the Viewer files are placed in a subfolder of IIS's `wwwroot` folder. Alternatively, you could create an IIS virtual directory that maps to the folder where the Viewer files are stored.

#### ▶ To install the HTML5 Viewer manually:

1. Create a folder in a location that is known to your web server.  
For example, you could create a folder named `HTML5Viewer` in the following location:  
`C:\inetpub\wwwroot\HTML5Viewer`
2. Extract the files from `Viewer.zip` to the new folder on your web server.  
`Viewer.zip` is provided in the installation package.

Resources	File folder	
Handheld.html	HTML Document	4 KB
Index.html	HTML Document	5 KB
proxy.ashx	ASP.NET Generic Handler	9 KB
proxy.config	XML Configuration File	2 KB
proxy.jsp	JSP File	5 KB
proxy.php	PHP File	6 KB
Tablet.html	HTML Document	5 KB
web.config	XML Configuration File	2 KB

You have installed an instance of the HTML5 Viewer. The next step is to verify the installation by launching the viewer.

### 7.2.2 Launch the Viewer

There is an HTML file in the deployment folder for each class of device that can be used to run the viewer:

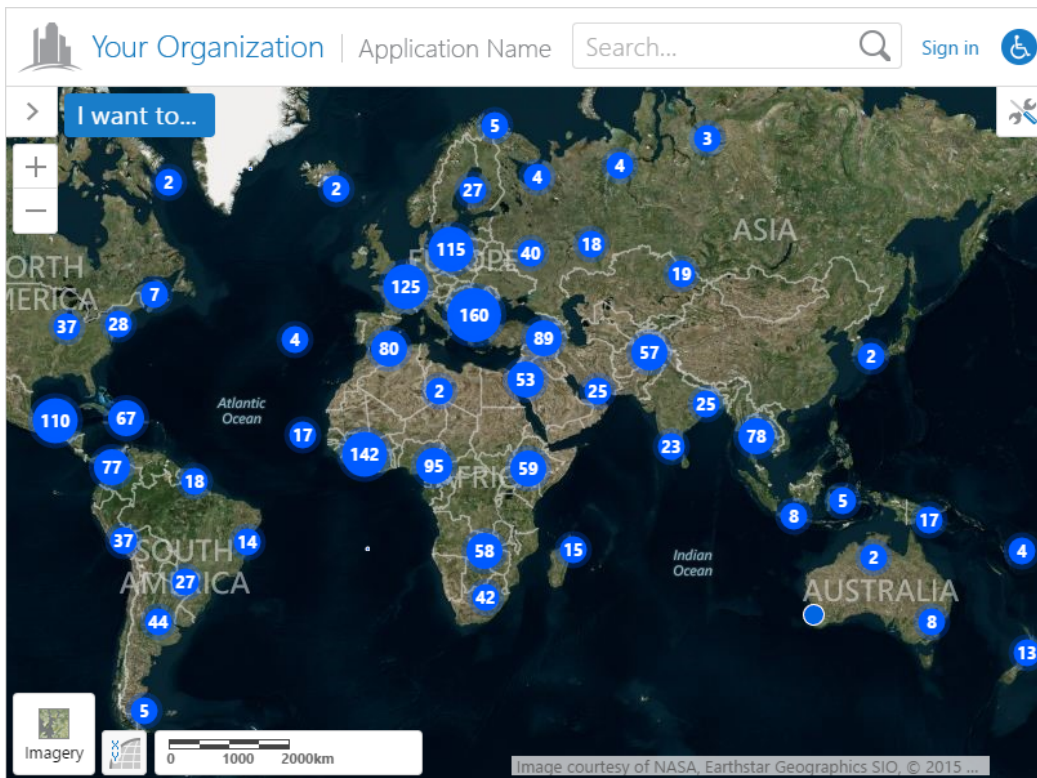
- **Index.html**: For running on desktop computers.
- **Handheld.html**: For running on handheld devices, such as smart phones.
- **Tablet.html**: For running on tablets.

By default, the HTML5 Viewer is configured to target a World Cities site on a sample Geocortex server.

► **To verify your installation:**

1. Launch the viewer in your browser using the Desktop interface.

For example, if you deployed the viewer to `C:\inetpub\wwwroot\HTML5Viewer`, type `http://localhost/HTML5Viewer/Index.html` in your browser's address bar.



Out-of-the-box HTML5 viewer showing a sample site in the Desktop interface

2. Launch the viewer in the Handheld interface.

The URL is `http://localhost/[deployment folder]/Handheld.html`.

3. Launch the viewer in the Tablet interface.

The URL is `http://localhost/[deployment folder]/Tablet.html`.

## 7.2.3 Point the Viewer to Your Site

The viewer has three configuration files, one for each class of device that can be used to run the viewer:

- `Desktop.json.js`: For running on desktop computers.
- `Handheld.json.js`: For running on handheld devices, such as smart phones.
- `Tablet.json.js`: For running on tablets.

To make the viewer point to your site rather than the sample site, you need to change the `siteUri` property in each of the viewer's three configuration files.

► **To configure the viewer to target your site:**

1. Open the three configuration files in a text editor.  
The configuration files are in [deployment folder]/Resources/Config/Default, where [deployment folder] is the folder on your web server where you deployed the viewer.
2. In each configuration file, update the `siteUri` property to point to your site. For example:

```
...  
"siteUri": "http://myserver.com/Geocortex/Essentials/REST/sites/MySite"  
...
```

3. Save the configuration files.
4. Test the three configurations by launching each one in a browser.

## 8 Upgrading

Upgrading installs the same components in the same location as the existing installation.

You can upgrade Essentials and the viewers independently—you can upgrade only Essentials, or only the viewers, or both Essentials and the viewers. To use certain features, you may have to upgrade both Essentials and the viewers. For best results, we recommend upgrading both Essentials and the viewers. To find out the Essentials version that a particular viewer version works with best, see the "Geocortex Essentials Requirements" section in the viewer's documentation.

Usually, your custom files such as viewers are upwardly compatible—after you upgrade the software, the old files still work. Occasionally, improvements to the software break upward compatibility. In this case, you may have to perform some manual steps to complete the upgrade. For information, see [Versions that Require Manual Steps on page 21](#).

### 8.1 Upgrade Options

The method that you use to upgrade the HTML5 Viewer depends on how you installed the viewer:

- **Installed Using a Viewer Template:** You can use Essentials to upgrade the viewer.
  - Read [About Upgrading Viewers Using Essentials on page 19](#).
  - Read [Versions that Require Manual Steps on page 21](#).
  - Follow the instructions in [Upgrade Viewers Using Essentials on page 24](#).
- **Installed Manually:** If you installed the viewer by dropping it directly into a web server:
  - Read [About Upgrading the Viewer Manually on page 21](#).
  - Read [About Upgrading the Viewer Manually on page 21](#) to see if you need to perform any manual steps.
  - Follow the instructions in [Upgrade the Viewer Manually on page 30](#).  
This upgrade procedure retains the manual nature of your deployment.

#### 8.1.1 About Upgrading Viewers Using Essentials

There are two main steps to upgrading the HTML5 Viewer using Essentials:

- **Upgrade the Template:** See [Step 1: Upgrade the Viewer Template on page 20](#).
- **Upgrade your Configured Viewers to Use the New Template:** See [Step 2: Upgrade Configured Viewers on page 20](#).



Upgrading the template removes the old template. This means that the viewers are unusable between the time you upgrade the template and the time you upgrade the configured viewers—they cannot be launched or edited during this time.

## Step 1: Upgrade the Viewer Template

To upgrade a viewer template, you use the Post Installer's Upgrade function. The Upgrade function partially automates the process of upgrading the viewer files that are deployed to the web server.

When you upgrade a viewer, you want to take advantage of the new features and improvements that the new template offers. At the same time, you want to preserve any changes that you have made to the viewer, such as customizations to HTML, CSS, JavaScript, or image files. The Upgrade function examines each file in your existing viewer so your customizations can be preserved:

- **Installed File Not Customized:** Viewer files that you have not customized are replaced when you upgrade. This means that you benefit from any changes that have been made in the new versions of these files.
- **Installed File Customized, Unchanged in New Template:** If you have modified a viewer file, but the file has not changed in the new release, then the installed version of the file is preserved. This preserves your customization.
- **Installed File Customized, Changed in New Template:** If you have modified a viewer file and the file has changed in the new release, the Upgrade function allows you to decide how to upgrade the file—keep the installed file, replace the installed file with the new version of the file, or merge the two versions together.

If the server has WinMerge installed, the Upgrade function uses WinMerge to show a side-by-side comparison of the two versions of the file. You can use WinMerge's merge functions to selectively merge the differences. If WinMerge is not installed, you must merge the files manually, by editing them and copying over the changes.



WinMerge is not installed with Essentials. If you want to use WinMerge to compare and merge files, you must install WinMerge before you perform the upgrade. You can download WinMerge for free from <http://winmerge.org/>.

- **New Custom Files:** If you have added new files to the viewer, upgrading preserves them. For example, if you have created custom modules, the files that implement the modules are not affected by upgrading—they remain in the same location, with the same names, as before the upgrade.

By default, the Upgrade function deploys the upgrade to the same location as the existing viewer. Deploying to the same location ensures that the viewer's URLs stay the same, so you do not need to update the launch links and URLs that you provide to users. You can change the location if you want. If you deploy the viewer to a different location, you will have to update the launch links and URLs that you provide to users.

The Upgrade function creates a backup of the existing viewer.

## Step 2: Upgrade Configured Viewers

After you upgrade the HTML5 template, you must upgrade the HTML5 viewers that are configured in your sites. You cannot run or edit the viewers until you have upgraded them to use the new template.

Manager has an Upgrade Viewers function that enables you to upgrade all your configured HTML5 viewers with a single click. Alternatively, you can upgrade viewers individually.



## 8.1.2 About Upgrading the Viewer Manually

To manually upgrade an HTML5 viewer, you can perform the following tasks:

1. Create a backup of your configured viewer.  
For a typical viewer deployment, you can back up the following folder:  
`C:\inetpub\wwwroot\Html5Viewer`
2. Deploy the new viewer.
3. Manually add any configuration changes from your files in the backed up copy of the `Resources\Config\` folder to the upgraded `Resources\Config\` configuration files.  
Because the structure of the JSON configuration changes in new versions of the viewer, you should not overwrite the entire configuration file. Use a tool like [WinMerge](#) to detect your custom configuration and changes to the configuration file structure.
4. Manually add any changes you made to your viewer's `Index.html`, `Handheld.html`, `Tablet.html`, and any other host files.
5. Copy any other required resources from your backed up `Resources\` subfolders to the newly deployed `Resources\` subfolders.
6. Remove the old viewer from the connections list in Internet Information Services (IIS) Manager.

## 8.2 Versions that Require Manual Steps

Depending on the version that you are upgrading from, you may have to perform one or more manual steps when you upgrade a viewer. For some versions, you do not have to perform any manual steps.

The list below lists the viewer versions that may require manual steps. Scan the list to determine which steps apply to you, if any. To find out more about a step, click the link. Note that you may have to perform more than one of the steps in the list.

Upgrading from the following versions of the Geocortex Viewer for HTML5 may require manual steps:

- **Versions Older than 2.6:** Upgrade currency formatting in charts.  
See [Upgrade the HTML5 Viewer from Version 2.5.2 or Older on page 21](#).
- **Versions Older than 2.5:** Upgrade custom offline content.  
See [Upgrade the HTML5 Viewer from Version 2.4.1 or Older on page 22](#).
- **Versions Older than 2.4:** Update viewer URLs and configure trusted domains.  
See [Upgrade the HTML5 Viewer from Version 2.3.3 or Older on page 22](#).

### Upgrade the HTML5 Viewer from Version 2.5.2 or Older

Starting in Geocortex Viewer for HTML5 2.6, you can configure the currency type when you select the Currency format for the chart's category, series source, and series aggregator. By default, the currency type is based on the location of the Essentials server. For example, if the server is in the USA, the currency type is US dollars.

If a chart's data is stored in a different currency than the currency of your server's locale, you must edit the chart and set the three new Currency settings to the correct currency type. This ensures that the viewer interprets the chart's data correctly. For information on setting the Currency format for a chart's category, see "Category Setup (Horizontal Axis)" in the *Geocortex Essentials Administrator Guide*. For information on setting the Currency format for a series source and series aggregator, see "Series Setup (Vertical Axis)" in the *Geocortex Essentials Administrator Guide*.

## Upgrade the HTML5 Viewer from Version 2.4.1 or Older

As of version 2.5, the Geocortex Viewer for HTML5 cache manifest no longer exists. Instead, all content within the following folders will be made available offline in the Geocortex Mobile App Framework:

- The folder where the viewer is hosted.

For a typical HTML5 viewer deployment, the viewer is hosted in the following location:

```
C:\inetpub\wwwroot\Htm15Viewer\
```

- Essentials' virtual directory for the viewer.

For a typical installation, the virtual directory for an HTML5 viewer is located at the following location:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\REST  
Elements\Sites\MySite\Viewers\MyViewer\VirtualDirectory
```

Where **MySite** and **MyViewer** are your site name and viewer name.

If you have customized your cache manifest file, ensure all of your content is within these two folders so it will be available in the Geocortex Mobile App Framework. The existing cache manifest is located in the viewer's virtual directory at:

```
Resources\Config\CacheManifestConfig.xml
```

After you have ensured all of your content has been made available, we recommend you delete the `CacheManifestConfig.xml` file for each viewer.

We recommend you delete all `.manifest.xml` files from the virtual directory for each of your viewers. For example:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\REST  
Elements\Sites\MySite\Viewers\MyViewer\VirtualDirectory\MyViewer.manifest.xml
```

## Upgrade the HTML5 Viewer from Version 2.3.3 or Older



This section applies to you only if you are upgrading from a version of the viewer older than 2.4 and you have not performed the steps for [Geocortex Viewer for HTML5 Security Update 2015-03-26](#).

The viewer upgrade process replaces the viewer's `Framework.js` file with a new, secure version. After you have upgraded the viewer using the Post Installer, you may have to perform one of the following tasks to complete a successful upgrade to Geocortex Viewer for HTML5 2.9:

- If you use the `configBase` parameter in your viewer URLs, update the URLs to use fully qualified domain names. If you have not updated your viewer URLs, the URLs may not work in HTML5 viewers version 2.4 or newer. Follow the instructions in [Load Configuration from the Same Domain on page 23](#).

- If you use CORS to allow cross-origin resource sharing between Essentials and your viewers, configure the viewers' trusted domains.

If you have not configured trusted domains for your viewers, the viewers may not work in HTML5 viewers version 2.4 or newer. Follow the instructions in [Load Files from Another Domain on page 24](#).

## Load Configuration from the Same Domain

HTML5 viewers 2.4 and newer do not load configuration files that are hosted on the same domain as Essentials if the URLs have different origins. For example, the following URL fails to launch an HTML5 viewer version 2.4 because the viewer's configured domain, `https://myserver`, does not match the configuration's domain, `https://myserver.mydomain.com`, even though they map to the same machine:

```
https://myserver
/Html5Viewer/Index.html?configBase=https://myserver.domain.com/Geocortex/Essentials/...
```

If your environment is configured to produce URLs with two or more domains that do not match, you must update the URLs's configuration to work with new versions of the HTML5 viewer. We recommend that you use [fully qualified domain names](#) in your URLs, even if you do not use CORS or a proxy.

If your Essentials installation is configured for CORS, or your server is behind a reverse proxy, you must update the configuration. See [Set Up Cross-Origin Resource Sharing \(CORS\) on page 34](#) and "Run Essentials Behind a Reverse Proxy" in the *Essentials Installation Guide*.

You can configure the URLs that viewers and REST endpoints use from the viewer's `RestManagerAppSettings.xml` file and Essentials' `web.config` file. Configuring these URLs also updates the viewer launch links that appear in Manager.

### ► To update the viewer launch links in Manager:

1. Update the HTML5 viewer's base launch link:
  - a. Run an XML editor or text editor as an administrator.  
You can use Notepad, a text editor included with Windows.
  - b. Open the `RestManagerAppSettings.xml` file in the editor.  
For a typical Essentials installation, the file is located here:
 

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Manager\AppData\RestManagerAppSettings.xml
```
  - c. Within the viewer's `ViewerFramework` element, update the `Url` attribute to use a fully qualified domain name. For example:

```
<ViewerFramework ProductID="..." TemplateID="Html5Viewer_2_9"
Url="http://myserver.mydomain.com/Html5Viewer" />
```

- d. Save the file.

## 2. Update the Essentials URL:

- a. Open Manager's `web.config` file in the editor.

For a typical Essentials installation, the file is located here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\REST
Elements\Manager\web.config
```

- b. Within the `appSettings` element, find the `add` element for the `EssentialsUrl` key.
- c. Update the `value` attribute to use a fully qualified domain, for example:

```
<add key="EssentialsUrl"
value="http://myserver.mydomain.com/Geocortex/Essentials/REST/sites" />
```

- d. Save the file.

## Load Files from Another Domain

Starting in version 2.4, if you use Essentials is configured for CORS, you must configure the viewer with a trusted domain where Essentials is hosted. The trusted domains list is configured in each of a viewer's host files (`Index.html`, `Handheld.html`, and `Tablet.html`). You must configure the trusted domains for each viewer. For instructions, see [Configure Your Essentials Domain in Your Viewers](#).

## 8.3 Upgrade Viewers Using Essentials



If you installed the HTML5 Viewer framework manually, you cannot use this method to upgrade the viewer. See [Upgrade the Viewer Manually on page 30](#).

### 8.3.1 Step 1: Upgrade the Viewer Template

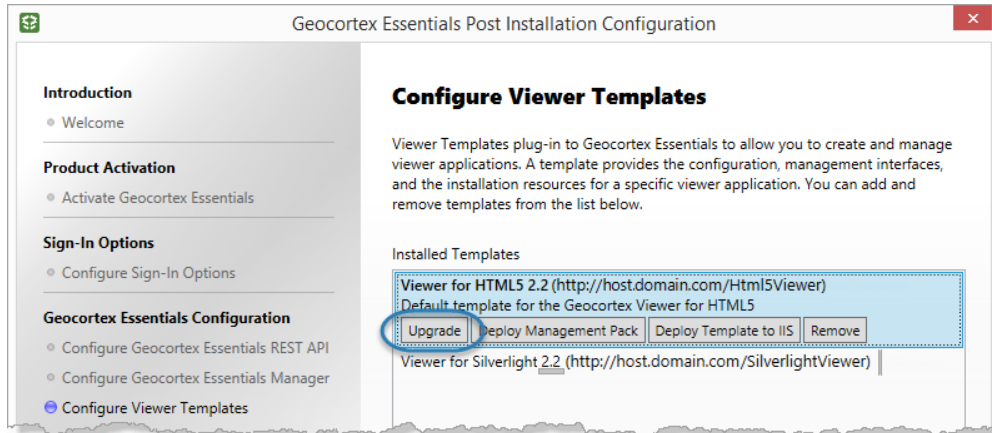
#### ► To upgrade the HTML5 Viewer template using Essentials:

Before you begin, you must download the installation package for the version of the viewer framework that you want to upgrade to. See [Download the Installation Package on page 8](#) for instructions.



Upgrading a viewer template removes the existing template.

1. In the Post Installer, click **Configure Viewer Templates** in the side panel.
2. In the **Installed Templates** area, click the template that you want to upgrade. The template management buttons show.



### Location of the Upgrade button for upgrading a viewer template

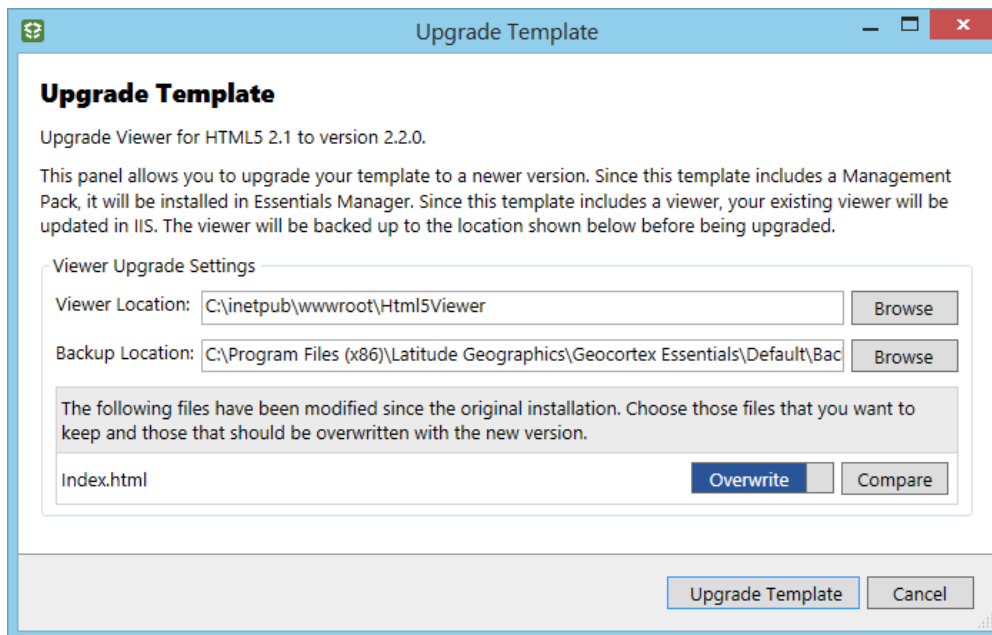
3. Click **Upgrade**.
4. Browse to the location where you extracted the installation package.



If a dialog appears that says **If you have deployed a viewer for this template, please select its location in the next step. If not, just press Cancel.** and you have previously deployed a viewer for the template you are attempting to upgrade, press OK and browse to the deployed viewer. By default, this will be inside the `C:\inetpub\wwwroot\` folder.

5. Select the template file, `Geocortex.Essentials.HTML5Viewer.Template.[version].vte`, and then click **Open**.

The Upgrade Template dialog box opens.

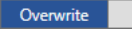



### Example of the Upgrade Template dialog box

6. **Viewer Location:** If the folder location given in the Viewer Location box is not where you want to deploy the upgrade, click **Browse**, select the folder where you want to deploy the upgrade, and then click **OK**.  
By default, the Viewer Location box shows the location where the viewer was installed. If you moved the viewer after installing it, you will have to change the location.
7. **Backup Location:** If you want to change the folder where the upgrader puts the backup of the old viewer, click **Browse**, select the folder where you want to put the backup, and then click **OK**.
8. **Modified Files:** If there are modified files, review each file and indicate whether you want to keep the existing version of the file or overwrite the existing file with the new version.  
A file appears in the list only if there are software updates in the new version of the file, and you have customized the existing version of the file. This step gives you the opportunity to merge your modifications into the new file before the upgrader overwrites the installed file, thus preserving your customization.



Alternatively, you could merge the software updates into the existing file and use the Keep Existing action. The easiest approach is to merge into whichever file has the most changes.

- a. Merge your modifications to the new file using one of the following methods:
    - If you have **WinMerge** installed, click the **Compare** button to review the differences between the two files in WinMerge, and then use WinMerge's **Merge** function to merge your customization into the new file. Save your merges.
    - If you do not have WinMerge installed, perform your merges manually by copying and pasting your customizations into the new file using a text editor. Save the file.
  - b. In the Post Installer, set the file's action to **Overwrite** .  
Overwrite replaces the installed version of the file with the new version.  
To change the action from Keep Existing to Overwrite, click the **Keep Existing** button .
9. Click **Upgrade Template**.  
The old template is removed and the new template is installed.
  10. When the success message displays, click **OK**.  
The Upgrade Template dialog box closes. You have completed the template upgrade.

You are ready to upgrade the HTML5 viewers that are configured in your sites. Click **Finish** and follow the prompts to close the Post Installer.

### 8.3.2 Step 2: Upgrade your Configured Viewers

You can upgrade all your configured HTML5 viewers at one time (instructions below), or upgrade them individually (see [To upgrade a single viewer: on page 27](#)).

► **To upgrade all of your configured viewers at one time:**

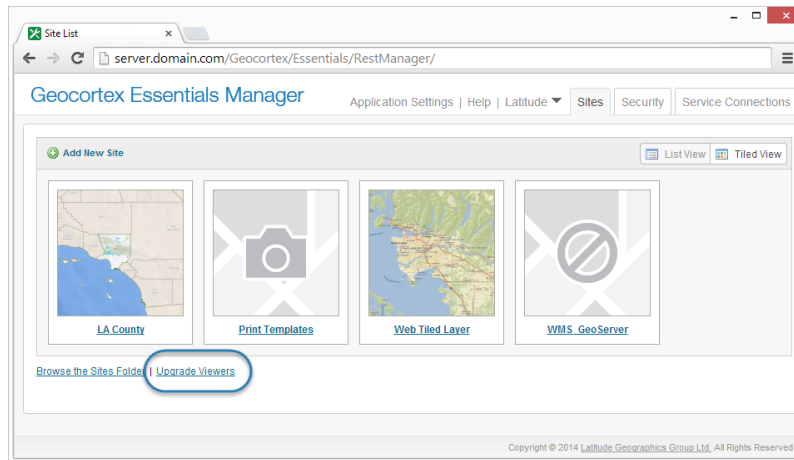
1. Do one of the following:

**If Manager is Closed**

1. Launch Manager.  
You will be prompted to upgrade your viewers.

**If Manager is Open**

1. In Manager, choose the **Sites** tab. If another site is open for editing, press **Save Site** and then **Close Site** to return to the Sites list.
2. Click the **Upgrade Viewers** hyperlink on the **Sites** tab.



**Location of the Upgrade Viewers hyperlink**

2. Make sure the HTML5 Viewer checkbox is selected and click **OK**.



If you click Cancel instead of OK, you can upgrade the viewers later using the Upgrade Viewers hyperlink on the Sites tab.

3. When the **Done** message shows, click **Close**.  
The Upgrade Viewers dialog box closes. You have completed upgrading your HTML5 viewers. You can now run your viewers and edit them in Manager.

► **To upgrade a single viewer:**

1. If Manager is closed, launch Manager and click **Cancel** when you are prompted to upgrade your viewers.
2. When you are ready to upgrade a viewer, edit the viewer.  
You will be prompted to upgrade the viewer.
3. Click **Upgrade**.  
Manager upgrades the viewer.

## 8.4 Restore a Viewer

If you upgraded a viewer using Essentials, and then decide that you want to roll back to the older version, follow the instructions below to remove the new version and restore the old version.

### Viewer Backups

To restore a viewer, you must have a copy of the folder where the viewer was deployed in the web server. If you upgraded the viewer using the Post Installer's Upgrade function, a backup is made automatically. For an instance of Essentials called Default, viewer backups created by the Upgrade function are in the following folder:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\Backups
```

When the Post Installer's Upgrade function is used to upgrade a HTML5 viewer, a site's old configuration settings get backed up as well. You can find the configuration settings for the old viewer at this location:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\REST Elements\Sites\Your Site\Viewer\VirtualDirectory\Resources\Config\Default
```

The backup configurations have the file extension `.backup.YYYYMMDD`, where `YYYYMMDD` is the date that the backup was made.

### Compatibility Issues

A minor release may introduce breaking changes in the configuration settings. Incompatible configuration is automatically fixed by the software during the upgrade process, but it must be manually fixed if you are downgrading. To test your site's configuration for backwards compatibility, we recommend that you install both versions of the viewer, attempt to view your site through both of them, and compare the configuration settings side by side.

You can use a tool like [DiffChecker.com](http://DiffChecker.com) to visualize the configuration changes.

#### To restore a previous version of a viewer:


1. In the file system, locate the folder that contains the backup of the viewer that you want to restore.  
If you upgraded the viewer using the Post Installer's Upgrade function, the backups are in the following folder:

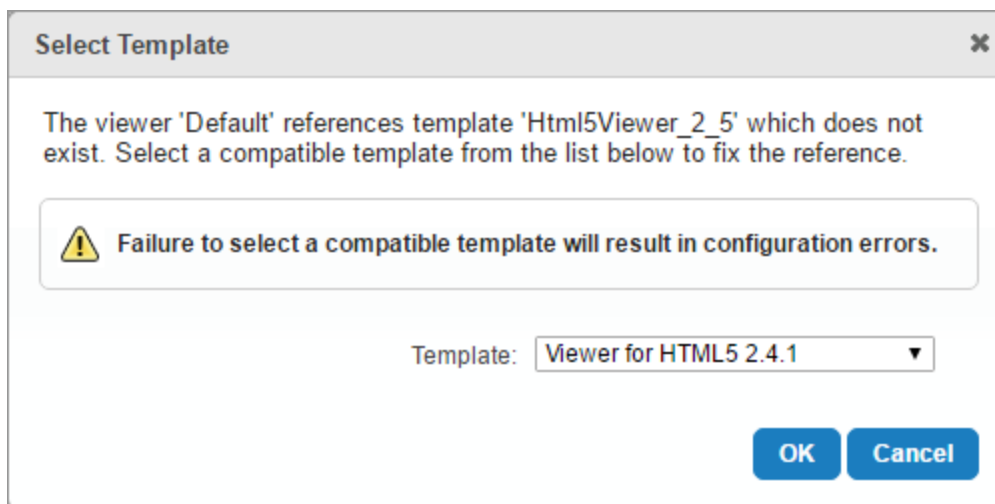
```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\Backups
```

The folder `Default` name may be different depending on the name of your instance of Essentials.

2. Download the old template from the Geocortex Support Center (<https://support.geocortex.com/>).  
See [Download the Installation Package on page 8](#) for instructions.



3. Launch the Post Installer.
  - **Windows Server 2012 or Windows 8, and Newer Versions:**  
On the **Start** screen, type **Post Install**, and then click **Post Installer**.
  - **Windows Server 2008 or Windows 7, and Older Versions:**  
In the **Start** menu, click **All Programs | Latitude Geographics | Geocortex Essentials [Version] [Instance] | Post Installer**.  
[Version] is the Essentials version number. [Instance] is the instance name, if Essentials is installed as a named instance. The default installation does not have an instance name.
4. Select **Configure Viewer Templates** in the menu.
5. In the **Installed Templates** area, click the template that you want to roll back.  
The template's management buttons are displayed.
6. Click **Remove**, and then click **Yes** to confirm.  
The removed viewer remains in the IIS `webroot` directory after this operation. If you wish to deploy to the same folder, you can go into the file system and manually rename or remove the viewer folder.
7. Install the old template.  
See [Install the Viewer Framework Using a Viewer Template on page 10](#) for instructions.
8. Copy the backup of the old viewer to the folder in the web server where you just deployed the viewer template.
9. Reload Manager and navigate to the Viewers page.
10. Ensure the viewer Launch URL matches the name of the folder where you just deployed the viewer template.  
If you deployed the old template to the same folder, the viewer may still reference the deleted template. In this case, the viewer does not display a Launch URL. To change the template the viewer references, select the  **Edit** icon next to the viewer to launch the Select Template dialog box and point the viewer to your preferred viewer template.



The Select Template dialog

11. Revert to your site's backed up viewer configuration files.

If you upgraded your viewer template using the upgrader in the Post-Installation Configuration, the old versions of your viewer configuration files are backed up in the configuration folder. For a typical installation of Essentials, you can locate the configuration folder at this location:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex
Essentials\Default\REST Elements\Sites\Your
Site\Viewer\VirtualDirectory\Resources\Config\Default
```

The backup configurations have the file extension `.backup.YYYYMMDD`, where `YYYYMMDD` is the date that the backup was made.

- a. Move any active `.json.js` configuration files to another folder on your system for safe keeping. This may include files with `Preview.json.js` in the file name.
- b. Remove the `.backup.YYYYMMDD` suffix from the remaining files.

For more information, see [Viewer Backups](#).

12. Open the viewer in a web browser and verify that it works.

## Troubleshooting

If your rolled back viewer does not open or states that the viewer configuration is invalid or malformed, we recommend examining the configuration of the viewer and a backup configuration of the site before it had been upgraded. If you upgraded an HTML viewer template using the Post Installer's Upgrade function, you can compare your new configuration with the backup copies of your old configuration files. See [Viewer Backups](#) for information.

You can also generate a new viewer using the template that has just been installed. If the new viewer launches properly and the old one does not, compare the two viewer configurations to determine what the problem is. You can use a tool like [DiffChecker.com](#) to visualize the configuration changes.

## 8.5 Upgrade the Viewer Manually

This information is for users who are upgrading a manual installation of the Geocortex Viewer for HTML5. The instructions given here retain the manual nature of your deployment—this procedure does not integrate your viewers with Essentials.

### To upgrade a manual installation of the HTML5 Viewer:

1. Download the installation package for the new version of the viewer.  
Follow the instructions in [Download the Installation Package on page 8](#).
2. Remove the current Viewer from IIS.
  - a. Launch Internet Information Services (IIS) Manager.
  - b. In the **Connections** panel, expand the hierarchy and select the website where the HTML5 Viewer is deployed.  
By default, the Viewer is deployed to the Default Web Site.
  - c. In the **Actions** panel, click **View Applications**.
  - d. Right-click the HTML5 Viewer's application and select **Remove**.
  - e. When you are prompted to confirm, click **OK**.

- f. Close IIS Manager.
  - g. In **Windows Explorer**, navigate to the `wwwroot` folder in the `inetpub` folder.  
By default, the HTML5 Viewer's folder is in `C:\inetpub\wwwroot`.
  - h. Rename the HTML5 Viewer's folder.  
Keep the renamed folder as a backup.
3. Delete the contents of the deployment folder.
  4. Extract the files from `Viewer.zip` to the deployment folder.  
`Viewer.zip` is provided in the installation package.
  5. Update the `siteUri` element in all three viewer configuration files to point to the correct site.  
See [Point the Viewer to Your Site on page 17](#).
  6. Save the configuration files and test them by launching each of the layouts in a browser.  
See [Launch the Viewer on page 16](#).
  7. Manually copy your configuration changes from the backups to the new configuration files.
  8. Save and test the configurations.
  9. If you changed any other viewer files, such as HTML, CSS, JavaScript, or image files, copy the changes from the backup files to the new deployment.
  10. Save the files and test them.

## 9 Set Up a Proxy Page

There are two reasons to use a proxy page with the HTML5 Viewer:

- To allow **large requests**.  
Proxies permit large requests to be posted using HTTP POST, which avoids the size limitation on HTTP GET operations. This can be useful for editing and query operations, which can involve large requests.
- To enable **cross-origin access** to resources such as workflows or a geometry service.  
For accessing resources such as workflows and geometry services, using a proxy page is an alternative to cross-origin resource sharing (CORS). If you need to support legacy Internet Explorer browsers, note that Internet Explorer 8 and Internet Explorer 9 do not fully support CORS, so a proxy page would be required for cross-origin access.  
To enable cross-origin access when the viewer is deployed to different domain than Essentials, use cross-origin resource sharing. For more information, see [Set Up Cross-Origin Resource Sharing \(CORS\) on page 34](#).

The HTML5 Viewer installation package includes proxy pages for ASP.NET, Java (JSP), and PHP. These proxy pages are based on the ones that Esri provides for the ArcGIS API for JavaScript. For more information, refer to [Using the Proxy](#) in Esri's ArcGIS API for JavaScript documentation.

## 9.1 Use the ASP.NET Proxy Page that Ships with the Viewer

### To configure the ASP.NET proxy page to work with your HTML5 viewers:

These instructions assume that you deployed the HTML5 Viewer template to IIS.

1. If ASP.NET 2.0 or newer is not installed on the server where the Geocortex Viewer for HTML5 is deployed, install it now and register it with IIS.

The ASP.NET proxy page requires ASP.NET 2.0 or newer.

2. If you want to deploy the proxy page to a custom location:
  - a. Move or copy the `proxy.ashx` and `proxy.config` files to the desired location.  
By default, `proxy.ashx` and `proxy.config` are in the folder where you deployed the Viewer. The location that you copy the proxy files to must be known to your web server.
  - b. In IIS Manager, convert the folder containing `proxy.ashx` and `proxy.config` to an application that uses a .NET 2.0 or newer application pool.
  - c. For each HTML5 viewer that you have added to a site, configure the proxy's URI using one of the following methods:
    - In Manager, edit the viewer. In the side panel, click **Application**. In the **Proxy URI** box, type the location of your proxy. Click **Apply Changes** and **Save Site**.
    - Alternatively, open the viewer's three configuration files in a text editor and update each file's `proxyUri` element to point to the proxy location.

3. Run a text editor as an administrator and edit `proxy.config`.

The `proxy.config` file is used to configure the ArcGIS Server services that the proxy will forward to. By default, the `proxy.config` file is in `C:\inetpub\wwwroot\Html5Viewer`.

4. Add a `serverUrl` element for each of your servers.

Alternatively, modify any of the default `server.url` elements that you do not need.

You must have a `serverUrl` element for each geocoding service that you use. For information on configuring geocoding services, see "Geocoding Services" in the *Geocortex Essentials Administrator Guide*.

The `serverUrl` element has the following attributes:

- `url`: The location of the server.
- `matchAll`: Set to `true` to forward every request that begins with the `url`.
- `token`: (optional) The security token to include for a secured service.
- `dynamicToken`: Set to `true` to get the token dynamically using the user name and password that are stored in the `appSettings` section of the server's `web.config` file.



By default, the `ProxyConfig` element's `mustMatch` attribute is set to `true`. This prevents viewer access to any address that is not specified in the `serverUrls`.

5. Save the `proxy.config` file.

6. If your site accesses services that are secured using Windows Authentication, follow the instructions in [Adapt the ASP.NET Proxy Page to Access Windows-Secured Services on page 33](#).

## 9.2 Adapt the ASP.NET Proxy Page to Access Windows-Secured Services




We recommend you avoid securing your ArcGIS services with Basic Authentication because editing will not work if the ArcGIS services and the HTML5 Viewer are hosted on different servers; use Windows Authentication instead.

If your site contains ArcGIS services that are secured using Windows Authentication, the proxy must forward the credentials to the secured services.

### ▶ To adapt the ASP.NET proxy page to access Windows-secured services:

#### Step 1: Configure authentication in IIS

1. In IIS Manager, expand the **Connections** panel's folders to show the location where you deployed the Viewer.
2. Click the Viewer. By default, its name is **Html5Viewer**.  
The center panel shows the Features View.
3. In the **IIS** section, double-click the **Authentication** icon .  
The center panel shows the authentication settings.
4. Right-click **ASP.NET Impersonation** and select **Enable**.



Some versions of IIS call this option "Windows Impersonation" instead of "ASP .NET Impersonation". If your version of IIS does not have an ASP .NET Impersonation option, enable Windows Impersonation.

5. Right-click **Windows Authentication** and select **Enable**.  
Do not close IIS Manager yet.

#### Step 2: Configure the proxy to forward the credentials

1. In IIS Manager, right-click the Viewer and click **Explore**.  
Windows Explorer will open to the folder that contains the proxy page.
2. Open the `proxy.config` file in a text editor.
3. Add a new `serverUrl` element.
4. Add a `url` attribute to the new `serverUrl` element and set the attribute to the URL of your Windows-secured server.
5. Add a `matchAll` attribute to the new `serverUrl` element and set it to `true`.

6. Add a `sendCredentials` attribute and set it to `true`.

The markup should look like the following, with `server.domain.com` replaced by your host name.

```
<serverUrl url="http://server.domain.com/arcgis/rest/services/"  
          matchAll="true" sendCredentials="true"></serverUrl>
```

7. Save the file and close the editor.
8. Close IIS Manager.

## 10 Set Up Cross-Origin Resource Sharing (CORS)



Configuring CORS via Internet Information Services (IIS) Manager is still supported. However, if Essentials and a viewer are hosted on separate domains and you require the Geocortex Mobile App Framework, you must use the following method to configure CORS safely. If you configure CORS in both Essentials and IIS Manager, CORS functionality will break.

If Essentials and an HTML5 viewer run on different domains, they require [cross-origin resource sharing](#) (CORS) to be set up. For example, when your Essentials installation and a viewer are deployed to different domains, the viewer cannot load configuration from Essentials. This is because the browser does not allow cross-domain requests without CORS.

This article outlines how you can configure CORS for your Essentials installation. The configuration requires two steps:

1. Configure CORS in Essentials Manager.
2. Configure CORS for each viewer that requires access to Essentials.



If you require cross-origin access to another domain that hosts resources used by Essentials or the viewer, you can configure a proxy page. See [Set Up a Proxy Page on page 31](#).

### Browser Compatibility

Cross-origin resource sharing is not fully supported in Internet Explorer 8–10 or Opera Mini 8. If you need to support those browsers, you can configure a proxy page to allow cross-origin access. See [Set Up a Proxy Page on page 31](#)

### Recommendations

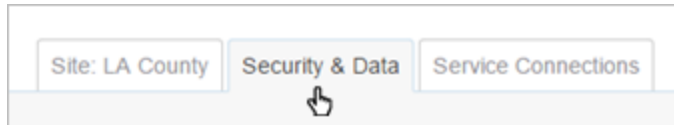
- Deploy Essentials and the viewer to the same domain to reduce management overhead and simplify Essentials upgrades.
- While the legacy method of configuring CORS is still supported, we recommend configuring CORS in Essentials Manager.

## Configure CORS in Manager



If you have previously configured CORS in Internet Information Services (IIS) Manager, you need to manually remove the old configuration settings before configuring CORS in Essentials Manager. For more information, see [Remove the Previous CORS Configuration from Essentials](#).

You can configure cross-origin resource sharing for Essentials in Manager. From the Security & Data tab in Manager's header go to the **Security > CORS** page to configure CORS.



### The Security & Data tab in Manager

In most Essentials environments, you can configure CORS by adding the URL where your viewers are hosted using the **Allowed Origins** form. You can add URLs as fully qualified domain names or IP addresses. For example, add an entry for `https://viewer.domain.com` if that is the domain your viewer is hosted on.

If your URLs use both `http://` and `https://`, add both forms of the full URL as separate entries.

 A screenshot of the "Allowed Origins" form. It features a text input field containing "http://server.domain.com/path/" and an "Add" button. Below this, there is a table with two rows. The first row contains "http://127.0.0.1:8181" and a trash can icon. The second row is empty.
 

Allowed Origins	
<input type="text" value="http://server.domain.com/path/"/>	<input type="button" value="Add"/>
http://127.0.0.1:8181	

### The CORS Allowed Origins form

If your CORS setup requires custom HTTP request methods and headers, you can also configure them from the Allowed Methods and Allowed Headers forms on this page. These settings only require configuration in custom environments that require additional HTTP headers. For more information, see [Allowed Methods and Allowed Headers Configuration](#).



The entry `http://127.0.0.1:8181` points at localhost and is required in order to use the Geocortex Mobile App Framework.

### Add or Remove Values

You can add an allowed value by entering it in the form's text box and pressing the **Add** button. Existing entries are displayed below the input form. To remove an entry you can use the **Remove** button next to each entry.

Once you have added values, use the Apply Details button to save the configuration.

## Configure Your Essentials Domain in Your Viewers

To complete CORS configuration, you must configure each of your viewers to use the domain name where Essentials is hosted.

### ► To configure Essentials' trusted origin in your viewers:

1. Run a text editor as an administrator.  
Notepad is a text editor included with Windows.
2. Open the default viewer host file in the editor.  
The default host page for HTML5 viewers is `Index.html`. For a typical HTML5 viewer deployment, the `Index.html` host file is located here:

```
C:\inetpub\wwwroot\Htm15Viewer\Index.html
```

3. Find where the `viewerConfig` variable is defined:

```
var viewerConfig = {  
    ...  
}
```

4. Immediately before the section where the `viewerConfig` variable is defined, configure the Essentials origin to be a trusted origin. Use the `geocortex._configDomains` object to configure trusted origins.  
For example, if Essentials is hosted at `http://myserver.mydomain.com`:

```
geocortex._configDomains = {  
    "http://myserver.mydomain.com": true  
};
```

5. Save the `Index.html` file.
6. Repeat Steps 1–5 for `Tablet.html`, `Handheld.html` and any other host files.
7. Repeat Steps 1–6 for each of your HTML5 viewers.

## Allowed Methods and Allowed Headers Configuration

In addition to the Allowed Origins forms, you can configure the Allowed Methods and Allowed Headers values for Essentials. The default values provided in these fields are sufficient for most Essentials environments.

- **Allowed Method:** Add or modify HTTP request methods that Essentials accepts from allowed origins. The default allowed methods for CORS are `GET`, `POST`, `HEAD`, and `OPTIONS`. Web browsers may not honor your additional configurations. See the W3C's [Method Definitions](#) specification for more information about methods.
- **Allowed Header:** Add or modify HTTP header fields that Essentials accepts from allowed origins. The default allowed headers for CORS are `Content-Type` and `X-Requested-With`. See the W3C's [Header Field Definitions](#) specification for more information about header fields.



## Cookies and Authorization Headers



In most cases, cookies and authorization headers are not a required configuration step. Please be aware that the following documentation applies to advanced deployments only.

In order to support CORS HTTP requests that include cookies and authorization headers, you must configure Essentials to use the `Access-Control-Allow-Credentials` CORS header.

### Configure Essentials

You must edit your Essentials's `RestAppSettings.xml` file.

For a typical Essentials installation, you can find this file at the following location:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\REST  
Elements\Sites\RestAppSettings.xml
```

Locate the opening of the `<Cors>` object in the file, and add `AllowCredentials="true"` to the XML tag as follows:

```
<Cors AllowCredentials="true">
```

With Essentials now accepting CORS requests with credentials, you can now configure your viewers to issue the CORS requests.

### Configure the Viewer to Make CORS Requests to Essentials

Before the viewer can make CORS requests to Essentials, it needs to be configured to make requests to the server running Essentials that you have enabled credentials for in the previous section.

In your viewer's `Index.html`, `Tablet.html`, and `Handheld.html`, add the following lines inside `<script>` tags at the bottom of the file:

```
<script>
  // Uncomment the below line to only use local copies of Esri API files. This is
  // done automatically in the Geocortex Mobile App Framework.
  //var geocortexUseLocalEsriApi = true;

  var viewerConfig = {
    "configurations": {
      "default": "Resources/Config/Default/" + shellName + ".json.js"
    },
    "viewerConfigUri": null
  };

  var enableCors = function(viewer,loader) {
    require(["esri/config"], function(esriConfig) {
      esriConfig.defaults.io.corsEnabledServers.push({
        host: "server.organization.com",
        withCredentials: true
      });
    });
  };

  new geocortex.essentialsHtmlViewer.ViewerLoader().loadAndInitialize({
    onInitialized: enableCors
  });
</script>
```

Where the `host` property points to the server to the Essentials instance that you have configured for cookies and authorization headers, and where the `withCredentials` property is set to `true`.

## Configure the Viewer to Make CORS Requests to Other Servers

You can duplicate the configuration that the viewer uses to make requests to Essentials for other servers that require CORS requests with credentials. In this case, assign the host to the URL of the server you need to make requests to. You must ensure that the server uses the `Access-Control-Allow-Credentials` CORS header and that it is set to `true`.

## Remove the Previous CORS Configuration from Essentials

If you have configured CORS for your Essentials installation in IIS Manager, you need to remove the previous CORS configuration before using the CORS settings in Essentials Manager.

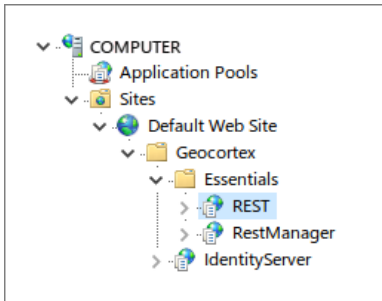
### ► To remove IIS Manager CORS configuration settings from Essentials:

1. Open Internet Information Services (IIS) Manager.
2. In the Connections panel, navigate to the site that hosts Essentials.
3. Locate the **REST** application.

For a typical Essentials installation, you can find the REST application within the following directory:

**Your Computer\Sites\Default Web Site\Geocortex\Essentials**

Once you have located it, select the REST application.



**Default location of the REST application**

4. In the REST application's Feature View, select the **HTTP Response Headers** section.  
Note any custom HTTP response headers you want to keep.
5. Remove every HTTP response header with following name:
  - `Access-Control-Allow-Origin`
  - `Access-Control-Allow-Methods`
  - `Access-Control-Allow-Headers`

To remove an HTTP response header, right-click it and select **Remove** from the context menu.

## 11 Architecture

### 11.1 About HTML5 Architecture

HTML5 opens the door to a number of new technologies that GIS professionals can use to deliver content to users. The new features of HTML5 provide an opportunity to create an entirely new class of rich, responsive, and intuitive web applications, making it extremely suitable for GIS purposes.

The Geocortex Viewer for HTML5 framework is designed to serve as a modern web development platform that provides organization, coherence, and stability. The framework is portable and maintainable, and it enables administrators and developers to configure and develop for both desktop and mobile browsers.

### 11.2 About HTML5 Applications

Geocortex HTML5 applications (viewers) are built on top of the Geocortex HTML5 framework. The Geocortex HTML5 framework uses an Application object to model an application in the browser.

Rather than use a multitude of global variables and methods to hold state and perform work, viewer applications hold their state and implementation in an instance of the Application object. Applications have a simple and well-defined life cycle.

Because of this, Geocortex HTML5 viewer applications are able to integrate well across a wide variety of platforms and existing applications.

### 11.3 Multi-Device Performance

Geocortex HTML5 viewers can run on different types of device (smartphones, tablets, or desktop computers) simply by altering the configuration and the basic way that UI components interact with each other.

Because of this ability, viewer applications should be written with performance and versatility in mind. For example, you should avoid expensive operations such as DOM querying or repeated HTTP requests. You should also avoid any task that might cause excessive work or that might prevent the user from interacting with the application.

Mobile devices come in many shapes and sizes, are not always connected to the Internet, and often run on batteries. It is therefore important to keep device and platform constraints in mind when developing and configuring HTML5 applications.

### 11.4 Modules, Views, and View Models

Geocortex HTML5 viewers are built with modules. Modules correspond to the features and functions that are available in viewers. For example, a viewer capable of displaying a list of map layers must include the SimpleLayerList Module. Similarly, for a viewer to run workflows, it must include the Workflow Module.

An HTML5 viewer can be thought of as a specific collection and configuration of modules. Modularization facilitates customization—modules can be removed, swapped, and reconfigured as needed to build the set of features and functions you want in your viewer.

To facilitate the development of modules, the Geocortex HTML5 framework uses a loosely coupled development style. Each module typically has no explicit dependencies on other modules, and the presence or absence of a particular module generally does not negatively affect the behavior of any other module.

The HTML5 viewer's user interface is built of components called views. Views are configurable user interface components associated with Modules that can be composed and rearranged at will.

See also...

[Model-View-ViewModel \(MVVM\) on page 415](#)

[UI Composition on page 418](#)

## 12 Viewer Launch URLs

### 12.1 About Viewer URLs

This section discusses URLs to launch an HTML5 viewer in a browser. For information on launching the Geocortex Mobile App Framework, refer to the *Geocortex Mobile App Framework Administrator Guide*.

URLs (Uniform Resource Locators) are also known as web addresses. For example, <http://www.esri.com/> is a URL.

A viewer URL is a URL that launches a viewer. In order for a user to launch a viewer, the user must know the viewer's URL, or have access to a hyperlink or QR code that links to the URL.



Manager provides viewer launch links in several places: in the Site List, on the Viewers page, and also on the Viewer Info page for a specific viewer. These links are useful while you are developing and testing a site.



For legibility, URLs and URL parameters are shown unencoded. You should always URL encode your URLs.

### Form of a Viewer's URL

The general form of a viewer's URL is:

```
http://<server.domain.com>/<iis-virtual-directory>/<page>?<url-parameters>
```

where:

- `<server.domain.com>` specifies the server that the viewer template is deployed to.
- `<iis-virtual-directory>` is the virtual directory in IIS that the viewer template is deployed to.
- `<page>` is the name of the page to launch, for example, `Index.html`.
- `<url-parameters>` are the parameters that allow you to control how the viewer launches—the extent that is initially shown, the language that is used, and so on.

The simplest URL to launch an HTML5 viewer is:

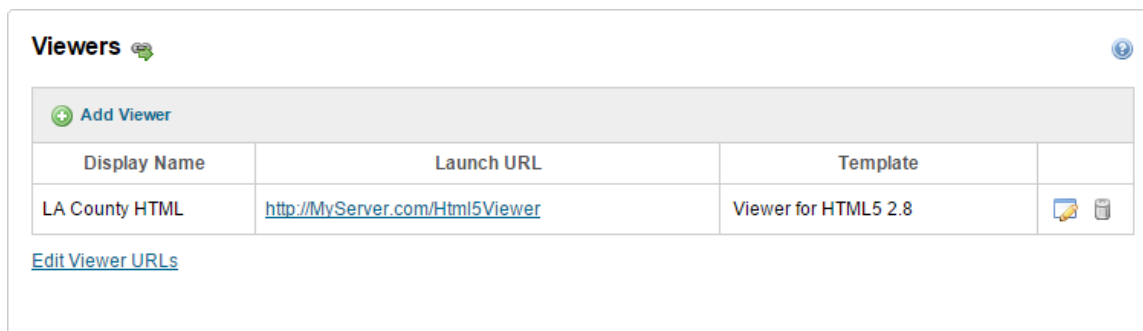
```
http://<server.mydomain.com>/<iis-virtual-directory>
```

This launches the default page, `Index.html`, for an unsecured site. The default page detects the type of device and launches the appropriate interface: Desktop, Tablet or Handheld.

To explicitly launch the Tablet interface for medium-format, touch-screen devices like tablets, use `Tablet.html` as the `<page>`. To explicitly launch the Handheld interface for small-format, touch-screen devices like smartphones, use `Handheld.html` as the `<page>`.

## Find the Launch URL of a Particular Viewer

When you add a Viewer in Geocortex Essentials, a launch URL appears in the **Viewers** section:



When you click this link or others like it in Manager, the viewer opens. However, the text that you see is a link and not the full URL of the viewer. To find the full URL, select and copy it from the address field in the browser. The full URL for the link above is actually:

```
http://  
MyServer.com/Html5Viewer  
/Index.html?configBase=http://MyServer.com/Geocortex/Essentials/REST/sites/LA_  
County/viewers/LA_County_HTML/virtualdirectory/Resources/Config/Default
```

## URL Parameters

A URL parameter is a string attached to a URL that specifies initial values or actions. For example, you can use the `extent` URL parameter to make the map zoom to a particular extent when the viewer is launched. From the end user's point of view, the URL parameter's value is the default value. For example, to the end user, the extent at which the map loads is the default extent.

To specify a URL parameter, you add a question mark to the end of the URL, followed by `parameter=value`. The general form is:

```
http://server.domain.com/vwr/Index.html?parameter=value
```

A URL can have multiple parameters. Parameters are separated by the ampersand character (&). The general form is:

```
http://server.domain.com/vwr/Index.html?parameter1=value1&parameter2=value2&...
```

## URL Parameters to Load Configuration Files

For a viewer to launch, the viewer must load a configuration file. If you know what type of device the end user will be using, you can specify which configuration file to load. If the viewer will be launched on different types of devices, you can let the viewer detect the device type and load the appropriate configuration file.

The viewer launch URL has two parameters that you can use to tell the viewer how to locate the configuration file to load:

- **configBase:** Specify the folder that contains the three different configuration files, and let the viewer detect the type of device and load the appropriate configuration file. The launch links in Manager use the `configBase` parameter.
- **viewerConfigUri:** Specify which configuration file to load. This parameter overrides the `configBase` parameter.

 The `viewerConfigUri` property has been deprecated.

### Example

In this example, the Handheld interface of the LA\_County site's LA\_County\_HTML viewer launches in Canadian French with debugging turned on. The site is not secured.


```
http://server.domain.com/vwr/Handheld.html?configBase=http://server.domain.com/Geocortex/Essentials/REST/sites/LA_County/viewers/LA_County_HTML/VirtualDirectory/Resources/Config/Default&locale=fr-CA&debug=true
```



## Publish Viewer URLs

As the application administrator, you must provide users with a way to launch the viewer. There are different ways to do this:

- **URL:** Give the viewer's URL to users, so they can type the URL into their browser's address bar.
- **Hyperlink:** Put a hyperlink representing the viewer's URL on a website or other web-connected resource. Users click the hyperlink to launch the viewer.

 URL-encode your URLs.

- **QR Code:** Provide a QR (Quick Response) code  in a web or print resource. To launch the viewer, users scan the QR code using a mobile device, such as a smartphone or tablet.

 To display a QR code for a launch link in Manager, hover the mouse pointer over the QR code icon  beside the launch link—the icon will expand into a scannable QR code.

## 12.2 Short Viewer URLs

You can set up a short launch URL for a particular viewer that has the form:

```
http://MyServer.com/Html5Viewer/index.html?viewer=MyViewer
```

There are a few different ways to set up the value for the `viewer` parameter:

- [Configure the \*\*Global Unique ID\*\* by editing the viewer.](#)  
We recommend this method because it is both the easiest and the most robust in terms of upgrading the viewer.
- [Edit the \*\*ViewerSettings.json.js\*\* file.](#)  
This method may slightly improve performance but requires manually editing a viewer configuration file. Any changes will also survive after upgrading the viewer. This method overrides the previous method.
- [Edit the \*\*Index.html\*\* file.](#)  
This method offers the greatest performance but is also the most fragile. Any changes must be reapplied after upgrading the viewer. We do not recommend using this method. This method overrides the previous methods.

### ▶ To configure the **Global Unique ID** by editing the viewer:

1. In Manager, edit the viewer.
2. On the **Viewer Info** page, type any value without special characters for **Global Unique ID**. For example, **MyViewer**.



As the name suggests, the **Global Unique ID** should be unique for each viewer across all sites and instances of Essentials.



The **Global Unique ID** must not contain special characters such as `<`, `>`, `%`, `&`, `:`, `*`, `\` / and `?`.

3. Click **Apply Changes**.
4. Click **Save Site**.

The **Launch in Browser** link now contains the short launch URL in the form of:

```
http://MyServer.com/Html5Viewer/index.html?viewer=MyViewer
```

where the `viewer` parameter is the same value as the **Global Unique ID** of the viewer.



If you do not set the **Global Unique ID**, all viewers use the default Global Unique ID, which consists of the Site ID and the Viewer ID separated by a period character. For example,  
`http://MyServer.com/Html5Viewer/index.html?viewer=MySiteID.MyViewerID`

### ▶ To edit the **ViewerSettings.json.js** file:

1. In the root folder where the HTML5 Viewer is installed, open the `ViewerSettings.json.js` file in a text editor. The default location is:



C:\inetpub\wwwroot\Html5Viewer\ViewerSettings.json.js



You may need to run your text editor as an Administrator to modify this file.

2. For each viewer, do one of the following:

- To configure a short launch link for any interface, in the `viewerSettings` array, add a new object with the following properties:

```
{
  "viewerSettings": [
    {
      "id": "MyViewer",
      "url": "http://MyServer.com/Geocortex/Essentials/REST/sites/LA_
County/Viewers/LA_County_HTML/VirtualDirectory/Resources/Config/Default/",
      "securityUrl": ""
    }
  ],
  ...
}
```

Set `id` to any value without special characters you want to use for the `viewer` parameter.

Set `url` to the folder containing the viewer configuration files.

If the site is secured, set the optional `securityUrl` property to the Essentials sign-in URL. For example:

```
http://MyServer.com/Geocortex/Essentials/REST/security/signin?token_
type=fragment&app=
```

- To configure a short launch link for a specific interface such as the Handheld interface, in the `viewerSettings` array, add a new object with the following properties.



The `viewerConfigUri` property has been deprecated.

```
{
  "viewerSettings": [
    {
      "id": "MyViewer",
      "viewerConfigUri":
      "http://MyServer.com/Geocortex/Essentials/REST/sites/LA_County/viewers/LA_
      County_HTML/VirtualDirectory/Resources/Config/Default/Handheld.json.js",
      "securityUrl": ""
    }
  ],
  ...
}
```

Set `id` to any value without special characters you want to use for the `viewer` parameter.

Set `viewerConfigUri` to the location of a specific viewer configuration file.

If the site is secured, set the optional `securityUrl` property to the Essentials sign-in URL. For example:

```
http://MyServer.com/Geocortex/Essentials/REST/security/signin?token_
type=fragment&app=
```

3. If you want the viewer to use a [custom splash screen image](#), add the `splashScreenUrl` property to the `viewerSettings` object. Set `splashScreenUrl` to the image's fully qualified URL.



When you configure the short URL in the `ViewerSettings.json.js` file, you must configure the custom splash screen image in the `ViewerSettings.json.js` file instead of in Manager.

```
{
  "viewerSettings": [
    {
      "id": "MyViewer",
      "url": "http://MyServer.com/Geocortex/Essentials/REST/sites/LA_
County/Viewers/LA_County_HTML/VirtualDirectory/Resources/Config/Default/",
      "splashScreenUrl": "http://host.domain.com/resources/images/splash-
image.png"
      "securityUrl": ""
    }
  ],
  ...
}
```

4. Save the file.

You can use the following URL to launch the viewer:

```
http://MyServer.com/Html5Viewer/index.html?viewer=MyViewer
```

► **To edit the Index.html file:**

1. In the root folder where the HTML5 Viewer is installed, open the `Index.html` file in a text editor. The default location is:

```
C:\inetpub\wwwroot\Html5Viewer\Index.html
```



You may need to run your text editor as an Administrator to modify this file.

2. For each viewer, do one of the following:

- Within the `ViewerInitializationOptions` object passed into the `ViewerLoader`'s `loadAndInitialize()` method, add a new property to the `aliases` property:

```
new geocortex.essentialsHtmlViewer.ViewerLoader().loadAndInitialize({
  aliases: {
    default: "Resources/Config/Default/" + shellName + ".json.js",
    MyViewer: "http://MyServer.com/Geocortex/Essentials/REST/sites/LA_
County/viewers/LA_County_HTML/virtualdirectory/Resources/Config/Default/" +
shellName + ".json.js"
  }
});
```

Set the name of the new property to the ID you want to use for the `viewer` parameter.

Set the value of the new property to the location of the viewer configuration file. The `shellName` variable allows the viewer to load different configuration files for each type of interface: Desktop, Tablet or Handheld.



Viewers defined using this method override viewers defined using the following alternative method.

- Within the `viewerConfig` variable, add a new property to the `configurations` property:

```
var viewerConfig = {
  "configurations": {
    "default": "Resources/Config/Default/" + shellName + ".json.js",
    "MyViewer": "http://MyServer.com/Geocortex/Essentials/REST/sites/LA_
County/viewers/LA_County_HTML/virtualdirectory/Resources/Config/Default/" +
shellName + ".json.js"
  },
  "viewerConfigUri": null
};
```

Set the name of the new property to the ID you want to use for the `viewer` parameter.

Set the value of the new property to the location of the viewer configuration file. The `shellName` variable allows the viewer to load different configuration files for each type of interface: Desktop, Tablet or Handheld.



The configuration files must be on the same domain as the viewer.

3. Save the file.

You can use the following URL to launch the viewer:

```
http://MyServer.com/Html5Viewer/index.html?viewer=MyViewer
```

## 12.3 URL Parameters Reference



For legibility, URLs and URL parameters are shown unencoded. You should always URL encode your URLs.

### URL Parameters that can be used in HTML5 Viewer launch URLs

#### center

**Specifies the coordinates at which to center the map when the map image loads.** If you want to specify the coordinates using a different spatial reference than the map's, append the WKID that you want to use.

**Syntax:** `?center=x,y` or `?center=x,y,wkid`

**Example 1:** In this example, the map pans to center on the specified map coordinates when the viewer is launched.

```
http://MyServer.com/Html5Viewer/Index.html?center=-13176043.9862,4002474.5385
```

**Example 2:** In this example, the coordinates are given in the specified spatial reference (WKID 4326) instead of the map's spatial reference. When the viewer is launched, the map is reprojected to WKID 4326 and then pans to the specified coordinates.

```
http://MyServer.com/Html5Viewer/Index.html?center=-114.7993,30.9820,4326
```

**Example 3:** In this example, the `center` parameter is used with the `scale` parameter. The `center` parameter ensures that the map is centered over the desired features. The `scale` parameter ensures that the map is zoomed to a scale at which the features are visible.

```
http://MyServer.com/Html5Viewer/Index.html?
center=255421.6566,6250846.4361,3857&scale=9000
```

**Module:** [Map Module on page 246](#)

#### configBase

**Specifies the URI of the folder where the configuration files are stored.** The viewer detects the device type and loads the appropriate configuration file from the specified folder. Use a fully qualified URI.

This URL parameter is useful if the viewer will be launched on different device types.

You can force the viewer to load a particular configuration file even though it uses `configBase`. This is useful when you are developing and testing your viewer.

**Syntax:** `?configBase=uri`, where `uri` is the URI to the folder containing the configuration files

**Example 1:** This example launches the `LA_County` site's `LA_County_HTML` viewer and loads the appropriate configuration file for the device type. This is how you would use `configBase` in a production viewer.

```
http://MyServer.com/Html5Viewer/Index.html?
configBase=http://MyServer.com/Geocortex/Essentials/REST/sites/LA_
County/Viewers/LA_County_HTML/VirtualDirectory/Resources/Config/Default
```

**Example 2:** This example launches the `LA_County` site's `LA_County_HTML` viewer and loads the Handheld configuration file. This is how you can view the Handheld interface on your desktop computer when you are testing the viewer.

## URL Parameters that can be used in HTML5 Viewer launch URLs

```
http://MyServer.com/Html5Viewer/Handheld.html?  
configBase=http://MyServer.com/Geocortex/Essentials/REST/sites/LA_  
County/Viewers/LA_County_HTML/VirtualDirectory/Resources/Config/Default
```

**Module:** None

### debug

**Logs additional types of event to the viewer's log file, and lists commands executed and events fired in the console.** If you do not want additional logging turned on, omit the `debug` parameter from the URL.

**Syntax:** `?debug=true`

**Example:** This example turns on additional logging for the viewer.

```
http://MyServer.com/Html5Viewer/Index.html?debug=true
```

**Module:** [Log Module on page 245](#)

### extent

**Zooms the map to the specified map coordinates.** If you want to specify the coordinates using a different spatial reference than the map's, append the WKID that you want to use.

**Syntax:** `?extent=minX,minY,maxX,maxY` or `extent=minX,minY,maxX,maxY,wkid`

**Example 1:** In this example, the map zooms to the specified map coordinates when the viewer is launched.

```
http://MyServer.com/Html5Viewer/Index.html?  
extent=1441172.484,550015.006,1456136.010,537754.744
```

**Example 2:** In this example, the coordinates are given in the specified spatial reference (WKID 4326) instead of the map's spatial reference. The map zooms to the specified latitude/longitude coordinates when the viewer is launched.

```
http://MyServer.com/Html5Viewer/Index.html?extent=35.2468,-80.8718,35.2139,-  
80.8210,4326
```

**Module:** [Map Module on page 246](#)

### layerTheme

**Specifies the layer theme to use when the viewer launches.** You can specify the layer theme either by its ID or by its Display Name.

**Syntax:** `?layerTheme=MyLayerTheme`, where `MyLayerTheme` is either the ID or the Display Name of the layer theme.

**Example 1:** In this example, the viewer launches using a layer theme with the ID, `0`.

```
http://MyServer.com/Html5Viewer/Index.html?layerTheme=0
```

**Example 2:** In this example, the viewer launches using a layer theme with the Display Name, `Imagery`.

```
http://MyServer.com/Html5Viewer/Index.html?layerTheme=Imagery
```

**Module:** [LayerThemes Module on page 242](#)

## URL Parameters that can be used in HTML5 Viewer launch URLs

### locale

**Specifies the language to use for the viewer.** For this parameter to work, a language file for the specified language must exist and be configured in the viewer configuration. See [About Translating UI Text on page 410](#) for instructions.



By default, viewers display date/time and number data in the user's locale. The `locale` parameter overrides the end user's locale preference. See "Format Date/Time and Number Fields" in the *Geocortex Essentials Administrator Guide* for information.

**Syntax:** `?locale=xx-YY`, where `xx-YY` is the language tag of the language file



The language code for `xx` is from the [ISO 639-1 standard](#). The country code for `YY` is from the [ISO 3166-1 standard](#).

**Example:** In this example, the viewer launches in Canadian French, provided the `fr-CA` language file exists and is configured.

```
http://MyServer.com/Html5Viewer/Index.html?locale=fr-CA
```

**Module:** None—`locale` is an application-wide property

### project

**Specifies a project to load when the viewer launches.** The `project` parameter is used in the Project URL that the viewer displays to end users so they can share the project with other users.

Projects belong to the site of the viewer that created the project. A user cannot create a project in one viewer and then load the project in a viewer that belongs to a different site.

**Syntax:** `?project=id`, where `id` is the ID of the project to load

You can find out a project's ID by viewing the project's details on Manager's Data Store - Documents page. For information, see "Geocortex Data Store" in the *Geocortex Essentials Administrator Guide*.

Alternatively, you can get the project's ID from the Project URL that shows on the viewer's Share Project panel. See [Sharing Projects on page 308](#) for information.

**Example:** This example launches the `LA_County_HTML` viewer and loads the project with ID `45ca42175c554a1e82c9a63cb3d570b3`:

```
http://host.domain.com/Html5Viewer/Index.html?viewer=LA_County.LA_County_HTML&project=45ca42175c554a1e82c9a63cb3d570b3
```

**Module:** [Project Module on page 304](#)

### run or runWorkflow

**Runs the specified workflow(s), mapping URL parameters to workflow input arguments.** The `run` and `runWorkflow` URL parameters perform the same function.

If you specify more than one workflow to run, the workflows run concurrently. Each additional workflow must be separated by a

## URL Parameters that can be used in HTML5 Viewer launch URLs

comma (,).



Workflows specified with the `run` or `runWorkflow` parameters will conflict if they target the same container in the viewer. A similar problem occurs if multiple workflows attempt to set the map extent or a layer's visibility.

If the workflow has arguments, you specify the arguments the same way you would specify additional URL parameters (`name=value`, with `&` separators). You must use the argument name as the parameter name. If an argument is optional, you can include it in the URL, or omit it.

**Syntax:** These two forms run workflows that do not have arguments:

```
&run=workflowId1,workflowId2,... or
&runWorkflow=workflowId1,workflowId2,...
```

These two forms run the specified workflow with the specified arguments (`arg1`, `arg2`, ...).

```
&run=workflowId&arg1=value1&arg2=value2&... or
&runWorkflow=workflowId&arg1=value1&arg2=value2&...
```

**Example 1:** In this example, the URL that you make available to end users launches the viewer and runs the workflow with ID "StartupWorkflow". StartupWorkflow does not have any arguments.

```
http://MyServer.com/Html5Viewer/Index.html&runWorkflow=StartupWorkflow
```

**Example 2:** This example runs the FindParcel workflow. The FindParcel workflow finds the specified parcel, and then optionally runs a report using the parcel's information. The workflow's two arguments are:

- **pid:** (required) The ID of the parcel to find.
- **reportToRun:** (optional) The ID of a report that reports parcel information.

```
http://MyServer.com/Html5Viewer/Index.html&
run=FindParcel&pid=02503116&reportToRun=MailingLabels
```

**Example 3:** This example runs the same FindParcel workflow as the previous example, except the optional `reportToRun` argument is omitted. In this case, the workflow finds the specified parcel, but does not run a report.

```
http://MyServer.com/Html5Viewer/Index.html&run=FindParcel&pid=02503116
```

**Module:** [Workflow Module on page 392](#)

## scale

**Specifies the initial scale.** The `scale` parameter takes the scale's denominator as its value. If the map does not have the specified scale, the nearest available scale is used.

**Syntax:** `?scale=value`, where the scale is 1:value

**Example 1:** In this example, when the viewer launches, the map zooms to the scale nearest to 1:2654.7813.

```
http://MyServer.com/Html5Viewer/Index.html?scale=2654.7813
```

**Example 2:** In this example, the `scale` parameter is used with the `center` parameter. This ensures that the map is zoomed and panned to show the desired view.



## URL Parameters that can be used in HTML5 Viewer launch URLs

```
http://MyServer.com/Html5Viewer/Index.html?center=255421.6566,6250846.4361,3857&scale=9000
```

**Module:** [Map Module on page 246](#)

### viewer

**Specifies the ID of the viewer to launch.** The value to use for the `viewer` parameter generally matches the **Global Unique ID** of the viewer. However, there are alternative ways to set up the value to use for the `viewer` parameter. For more information, see [Short Viewer URLs](#).



As the name suggests, the **Global Unique ID** should be unique for each viewer across all sites and instances of Essentials.



The **Global Unique ID** must not contain special characters such as `<`, `>`, `%`, `&`, `:`, `*`, `\` / and `?`.



If you do not set the **Global Unique ID**, all viewers use the default Global Unique ID, which consists of the Site ID and the Viewer ID separated by a period character. For example,  
`http://MyServer.com/Html5Viewer/index.html?viewer=MySiteID.MyViewerID`

**Syntax:** `?viewer=MyViewer`

**Example:** In this example, the viewer whose **Global Unique ID** is `MyViewer` is launched by the following URL:

```
http://MyServer.com/Html5Viewer/Index.html?viewer=MyViewer
```

The **Global Unique ID** is set by editing the viewer, on the **Viewer Info** page.

**Module:** None

### viewerConfigUri



The `viewerConfigUri` property has been deprecated.

**Specifies the URI of the configuration file to load.** We recommend using a fully qualified URI. The configuration file must be on the same domain as the viewer. This parameter overrides the `configBase` parameter.

This URL parameter is useful if you know the type of device that the end user will be using. If you do not know the device type, use the `configBase` URL parameter instead.

**Syntax:** `?viewerConfigUri=uri`, where `uri` is the URI to the configuration file

**Example:** This example launches the LA\_County site's LA\_County\_HTML viewer using the Handheld interface:

```
http://MyServer.com/Html5Viewer/Index.html?viewerConfigUri=http://MyServer.com/Geocortex/Essentials/REST/sites/LA_County/viewers/LA_County_
```

## URL Parameters that can be used in HTML5 Viewer launch URLs

HTML/VirtualDirectory/Resources/Config/Default/Handheld.json.js



Because you are specifying which configuration file to load, you could use the corresponding HTML page—`Tablet.html` for `Tablet.json.js`, or `Handheld.html` for `Handheld.json.js`—instead of `Index.html`. This is optional.

**Module:** None

## 12.4 URLs for Secured Sites

You do not need a different URL to access a secured site—you can use a regular viewer URL, as described in [About Viewer URLs on page 41](#). However, you can improve a viewer's launch performance by using a URL that links directly to the sign-in page. A direct link skips the steps required for the viewer to determine whether the site is secured.

When you link directly to the sign-in page, you pass the viewer's URL as a parameter. This ensures that the user's browser can be correctly directed to the viewer after the user has authenticated.

The general form for a URL that links directly to the sign-in page is:

```
http://server.domain.com/Geocortex/Essentials/REST/security/signIn?[url-parameters]
```



For legibility, URLs and URL parameters are shown unencoded. You should always URL encode your URLs.

The table below lists the parameters that you can use in sign-in page URLs.

### URL Parameters for Sign-In Page URLs

#### app

**The URL of the web application (viewer) to launch after the user has authenticated.**

The `app` parameter is required. The URL specified in the parameter should be URL encoded. The URL can have URL parameters.

**Syntax:** `app=url`

#### idp

**The security provider to use for authentication.**

Some security providers are identified using a predefined value. Other security providers are identified by the security provider's unique issuer seed. For information, see [Find the Value for the idp URL Parameter](#) below.

The `idp` parameter is optional. If you omit the `idp` parameter, the user's browser redirects to a page that lists the security providers for that site, and the user selects the security provider to sign in with.

**Syntax:** `idp=security-provider-identification`

### URL Parameters for Sign-In Page URLs

#### token\_type

**Required in viewer URLs that link directly to the sign-in page.**

**Syntax:** token\_type=fragment


## Find the Value for the idp URL Parameter

The `idp` URL parameter identifies the security provider to use for authentication. The table below lists the values for different security providers. For Geocortex Identity Server and ArcGIS Online security providers, you need the security provider's issuer seed. Instructions for getting the issuer seed are given below the table.

### Values to Use for the idp URL Parameter

Security Provider	Value	Example
Windows Integrated	AD AUTHORITY	idp=AD AUTHORITY
Anonymous Access	urn:gcx:guest	idp=urn:gcx:guest
Geocortex Identity Server	urn:gcx:idp:[issuer-seed]	idp=urn:gcx:idp:2895BE5E-FA19-4731-82F1-33F5FE30F786
ArcGIS Online	urn:gcx:ags:[issuer-seed]	idp=urn:gcx:ags:b2e2357d-84e8-4406-94aa-e34b798d1a98

### ► To get the issuer seed for a security provider:

1. In Manager, click the **Security & Data** tab, expand the **Security** area in the side panel, and then click **Providers**.
2. Click the **Edit** icon  beside the security provider whose issuer seed you want to get.
3. In the **Advanced** area, select the issuer seed and copy it.
4. Click **OK** to close the dialog box.
5. Paste the issuer seed in the URL's `idp` parameter.

### Example: URL for an HTML5 Viewer and Windows-Secured Site

Suppose you have a site that is secured using the Windows Integrated security provider. The URL below performs the authentication using Windows Authentication, and then launches the HTML5 viewer called `MyViewer`.

```
http://server.domain.com/Geocortex/Essentials/REST/security/signIn?token_
```

```
type=fragment&idp=AD+AUTHORITY&app=http://server.domain.com/viewers/Index.html  
?Viewer=MyViewer
```

### Example: URL for an HTML5 Viewer and Site that is Secured Using ArcGIS Online

Suppose you have a site that is secured using ArcGIS Online. The URL below performs the authentication using ArcGIS Online, and then launches the HTML5 viewer called `mobile` in the Handheld interface.

```
http://server.domain.com/Geocortex/Essentials/REST/security/signIn?token_  
type=fragment&idp=urn:gcx:ags:9b567ddc-27ab-4321-b968-  
570df7b53bfe&app=http://server.domain.com/gvh/Handheld.html?Viewer=mobile
```

## 13 About Viewer Configuration

You can think of HTML5 Viewer features as corresponding to modules. Most of the configuration of an HTML5 viewer involves changing the configuration of [modules](#), and their [views](#) and [view models](#). HTML5 viewers also have some application-wide properties.

There are two methods of configuring HTML5 viewers:

- **Manager:** Some modules and application-wide properties can be configured using Manager, provided the HTML5 Viewer's Management Pack is installed. For information on installing the Management Pack, see [Install the Viewer Framework Using a Viewer Template on page 10](#).
- **Configuration Files:** All properties—application-wide, module, view, and view model—can be configured by editing the viewer's configuration files. To do the editing, use a text editor like Notepad, or a JSON editor.



Before you edit a viewer configuration file in a text editor, you must close or sign out of Manager. If you run Manager at the same time that you edit a configuration file, you risk losing or corrupting the viewer configuration.

Some modules are partially configurable in Manager—in this case, you can do some configuration in Manager and some configuration in the configuration files. Modules that can be (at least partially) configured in Manager have a tip at the top of the module's section in [Configuration Settings by Module on page 129](#).

Configuring in Manager has advantages—in addition to providing an easy-to-use interface, Manager enables you to configure all three [configuration files](#) simultaneously. See [Configure a Viewer in Manager on page 65](#) for instructions.

For information about configuring application-wide properties, see [Application-Wide Settings on page 58](#) and [Configure a Viewer in Manager on page 65](#).

### 13.1 Configuration Files

Each HTML5 viewer you deploy has three configuration files which correspond to user interfaces for different device types—desktop, tablet, and handheld. The configuration files are:

- `Desktop.json.js`: For applications that display on desktop monitors.
- `Tablet.json.js`: For applications that display on tablets.
- `Handheld.json.js`: For applications that display on smart phones.

By using multiple configuration files, the Geocortex Viewer for HTML5 is able to support multiple device types in a single web application.

The information in the configuration files is written in JSON, a lightweight, easy-to-read syntax for storing and exchanging data.

Configuration files are usually kept in the application's `Config` subfolder. If you are not sure where your configuration files are located, see [File Locations on page 448](#).

## 13.2 Structure of Configuration Files

Geocortex HTML5 viewers use JSON configuration files to control which [modules](#), [views](#), and [view models](#) to load, and how to arrange them. This determines which features and functions are available in the viewer, their behavior, and much of the look and feel.

The main sections in an HTML5 Viewer configuration file are:

- **version:** The version of the HTML5 Viewer that the configuration file is intended to work with. The version is for information only. The version is automatically updated when you upgrade to a newer version of the Viewer.



Attempting to run the viewer with a different version of the configuration files gives unpredictable results.

- **application:** Application-wide properties such as the locations of the proxy page and the geometry service. See [Application-Wide Settings on page 58](#) for more information.
- **defaultLibraryId:** The library where the application looks for a compiled resource, if a library is not specified in the request. See [Libraries and the Default Library ID on page 60](#) for information.
- **libraries:** An array containing the libraries that make up the application. See [Libraries and the Default Library ID on page 60](#) for information.
- **modules:** An array of modules in the application. See [Modules on page 61](#) for information.
  - **views:** An array containing the module's views. See [Views on page 61](#) for information.
  - **viewModels:** An array containing the module's view models. See [View Models on page 61](#) for information.

Within the configuration for a module, the properties are organized into three groups: module properties, view properties, and view model properties, in that order. This can be used to help find a property within the configuration file.

- **widgets:** An array of widgets used by the viewer. A widget is a view component, such as a checkbox form item, that can be re-used across modules.

### 13.2.1 Application-Wide Settings



Some of the settings in the `application` section of a configuration file can be configured using Manager. For information, see [Configure Application-Wide Settings on page 70](#).

The `application` section of a configuration file contains application-wide settings. Some of the properties are present in the factory configuration files. Other properties are added by Manager when you configure the viewer's site.

The `application` section can have the following properties:

- **proxyUri:** The URI to the [proxy page](#), if you are using one. By default, the proxy URI is set to the ASP.NET proxy page that ships with the HTML5 Viewer, `proxy.ashx?`. The default URI is terminated with a question mark because the viewer attaches parameters to the URI. The question mark is optional—the viewer will add the question mark if necessary.

If you do not want to use a proxy page, clear the Proxy URI setting. In Manager, go to the viewer's **Application** page and remove `proxy.ashx?` from the **Proxy URI** box. Alternatively, edit the configuration files directly and set `"proxyUri": ""`.

For instructions on setting up a proxy page, see [Set Up a Proxy Page on page 31](#).

- **allowUnsafeContent**: When this property is set to `false`, the viewer does not interpret HTML markup in feature data—the viewer displays the actual, uninterpreted markup, not what the markup represents. It does this by escaping the HTML tags and quotation marks. For example, instead of resolving an anchor tag, the viewer displays the tag itself, so users see something like `<a href="http://server.domain.com/content">` in the viewer.

If you do not trust the data in your site's layers, make sure `allowUnsafeContent` is set to `false`. For example, if you use one or more map services that are publicly available on the Internet, you have no control over the data so you may not fully trust it. This is particularly true for KML and GeoRSS layers.

By default, `allowUnsafeContent` is `false`.

When `allowUnsafeContent` is set to `true`, the viewer interprets HTML markup in feature data. This is useful if you know that the layer's features contain HTML in one or more attributes, and you trust the data.

Note that `allowUnsafeContent` only affects HTML markup in feature data—it does not affect URLs in feature data. HTML5 viewers always examine feature data for URLs that could point to malicious content, regardless of whether `allowUnsafeContent` is `true` or `false`. For more information, see [Protecting Against Malicious Code on page 407](#).

- **OfflineStorageSpaceMb**: Controls the requested amount of available persistent storage for browsers that support the file system API. This setting does not apply to the Geocortex Mobile App Framework.



This setting has limited usefulness, as the file system API is currently only supported by Chrome and Opera, is no longer being maintained, and support may be dropped in future versions.

- **geometryServiceUrl**: The URI of the geometry service to use for projection and other geometry operations. The HTML5 Viewer's Geolocation feature uses the geometry service specified here to project the user's location from latitude/longitude coordinates to map coordinates.



As of version 2.5, the `geometryServiceUrl` property of the HTML5 Viewer has been deprecated. To configure a geometry service, edit the site instead. If configured, the geometry service in the site overrides the one configured here. For more information, see "Geometry Services" in the *Geocortex Essentials Administrator Guide*.

- **mobileMode**: To specify this viewer should operate in mobile mode, set to `true`; otherwise, set to `false`. In the Handheld interface, the default is `true`. In the Desktop and Tablet interfaces, this property is omitted (and therefore `false`) by default.
- **oAuth2ClientId**: The App ID of the ArcGIS Online application that the viewer uses to access private ArcGIS Online content. When you configure a site to access private ArcGIS Online content, Essentials adds the App ID to the viewer's configuration files. This improves the performance of viewer loading. For more information, refer to "Set Up Access to Private ArcGIS Online Content" in the *Geocortex Essentials Administrator Guide*.
- **datumTransforms**: An array of datum transformations to use in operations that require projection, such as geocoding, measurement, and buffering.



This feature requires an ArcGIS Server 10.1 geometry service. The geometry service is configured in the site.

The HTML5 Viewer has several built-in datum transformations:

- RD New (28992)
- British National Grid (27700)
- Czech S-JTSK(102067)



When you configure one or more `datumTransforms`, the viewer does not use the built-in datum transformations—the built-in transformations are only used if `datumTransforms` is empty.

A `datumTransforms` item has the properties listed below. To specify a datum transformation using the well-known ID (WKID), configure `transformWkid`, `fromWkid`, and `toWkid`. To use the well-known text (WKT) instead of the WKID, configure `transformWkt`, `fromWkt`, and `toWkt`.

- **`transformWkid`**: The WKID of the datum transform to use.
- **`transformWkt`**: The WKT of the datum transform to use. This is ignored if `transformWkid` is also specified.
- **`fromWkid`**: The WKID of the spatial reference to project from.
- **`fromWkt`**: The WKT of the spatial reference to project from. This is ignored if `fromWkid` is also specified.
- **`toWkid`**: The WKID of the spatial reference to project to.
- **`toWkt`**: The WKT of the spatial reference to project to. This is ignored if `toWkid` is also specified.



You only need to configure a datum transformation in one direction—the viewer will perform a backwards transformation when necessary. For example, if you configure a transformation from WKID 28992 to WKID 4326, the viewer can also perform a transformation from WKID 4326 to WKID 28992.

For a list of transformation WKIDs, refer to Esri's [ArcGIS Geographic and Vertical Transformation Tables](http://resources.arcgis.com/en/help/main/10.2/003r/pdf/geographic_transformations.pdf) ([http://resources.arcgis.com/en/help/main/10.2/003r/pdf/geographic\\_transformations.pdf](http://resources.arcgis.com/en/help/main/10.2/003r/pdf/geographic_transformations.pdf)).

## 13.2.2 Libraries and the Default Library ID

A library is a collection of code, CSS, and HTML files that are combined into a single JavaScript file. A library's contents implement [modules](#), [views](#), and [view models](#). The `libraries` section of a configuration file lists the libraries included in the application.

Each library has an ID and URI. When an HTML5 viewer is launched, the configured libraries are downloaded from their URIs. This downloads all the modules contained in the libraries (and the module's views and view models).

All compiled resources are keyed with the library ID. When a programmatic request is made for a resource, the request can specify the library that contains the requested resource. If the request does not include a library ID, the requested resource is assumed to be in the default library. The default library is configured in the `defaultLibraryId` property.



---

Location-specific resources are also keyed with the library ID. The `libraries` section in a configuration file relates language tags (for example, `en-US` for US English) to their locale-specific resource files.

### 13.2.3 Modules

Modules implement the underlying processes that make the Viewer act in response to user actions and input—the "business logic" of the Viewer. You can think of modules as corresponding to HTML5 Viewer features.

Modules are loosely coupled to other modules—technically, they are independent of each other, but in practice a module may "require" another module to be useful. For example, to make the Identify feature (Identify Module) available to users, you need to supply a means for users to indicate when they want to perform an identify operation—perhaps a tool in the toolbar (Toolbar Module), an item in the I Want To menu (IWantToMenu Module), or a workflow (Workflow Module).

The `modules` section of a configuration file lists the modules used by the viewer for that user interface—Desktop, Tablet, or Handheld. Each module is loaded from the library that the module belongs to.

Modules usually contain a number of views and view models.

For information on module properties, see [Common Settings for Modules on page 129](#) and the relevant subsection of [Configuration Settings by Module on page 129](#). For example, for the Banner Module, see [Common Settings for Modules on page 129](#) and [Banner Module on page 134](#).

### 13.2.4 Views

Views implement user interface elements.

Modules can have any number of configured views associated with them. Configured views are ready to be displayed whenever their target regions become available. Modules can also programmatically create views (and view models) on the fly.

For information on view properties, see [Common Settings for Views on page 130](#).

### 13.2.5 View Models

View models mediate between modules and views, exposing properties, commands, and other aspects of modules so they can be used in views.

Modules can have any number of configured view models. Modules can also programmatically create view models (and views) on the fly.

For information on view model properties, see [Common Settings for View Models on page 130](#).

## 13.3 General Method for Manual Configuration

The easiest way to manually configure a viewer is to change existing properties in the configuration files. For most properties, all you have to do is replace the existing value with the new value.

The HTML5 Viewer's default configuration files include configuration for every module, view, and view model available in the software. If you add something to the configuration, you can pattern it after something that already exists.

For a property that is an array, such as the items in the I Want To menu, you can delete a menu item by deleting the entire item—delete everything from the opening curly bracket "{" to the closing bracket "}", inclusive, and the separating comma, if there is one. To add an item to an array, copy and paste an existing array item from "{" to "}" inclusive, add a separating comma if necessary, and change its properties.



Make sure every item in the array has a comma after it, except the last item.



When modifying an HTML5 viewer's configuration, use [Configuration Settings by Module on page 129](#) as a reference. It describes the properties for each module, view, and view model. You can also configure some application-wide properties, described in [Application-Wide Settings on page 58](#).

► **To manually modify a viewer's configuration:**

1. Open the [configuration file](#) in a text editor or an XML editor.
2. Search for the module you want to modify by name.  
Use [Configuration Settings by Module on page 129](#) as a reference. A module's name is given in the section title, for example, "IWantToMenu" is the name of the module that implements the I Want To menu.
3. Locate the properties you want to change.  
Make sure you are looking in the correct subsection of the module's configuration—module, view, or view model.
4. Replace the existing values with the new values.
5. Save the configuration file.
6. Launch or refresh the viewer using the appropriate configuration—Desktop, Tablet, or Handheld—to verify your changes.
7. Repeat steps 2 - 6 for each module.
8. Repeat steps 1 - 7 for each configuration file.

## 13.4 About Configuration Using Manager

### 13.4.1 About Configuring Multiple Interfaces

The Geocortex Viewer for HTML5 is designed to work on different device types—desktop computers, tablets, and handheld devices like smartphones. The way the viewer does this is by allowing you to configure multiple user interfaces, one for each device type.

The configurations for the different user interfaces are in separate files. When you configure a particular aspect of an HTML5 viewer in Manager, you can choose between configuring all three interfaces together, or configuring them separately. Configuring the interfaces separately allows you to make them different from each other—configuring them together makes their configurable settings identical.

In Manager, HTML5 viewer configuration is divided into pages of related settings—I Want To menu settings, look and feel settings, and so on. The decision to configure the user interfaces together or separately only affects the settings on the current page. This means you can configure the interfaces together on one page but individually on another page.

For example, you might want to always use the same banner (configure the interfaces together on Manager's Look and Feel page), but have a different I Want To menu for handheld devices (configure the interfaces separately on Manager's I Want To Menu page).



If you have configured the interfaces separately on a particular page in Manager, and then you change back to configuring the interfaces together, Manager prompts you to select which configuration you want to keep—you will lose any variations that exist in the other two interfaces.



It is always safe to switch from configuring the user interfaces together to configuring them separately—you never lose any changes. If you are not sure whether you want the interfaces to be the same or different, keep them the same by configuring them together—you can always configure them separately later.



The issue of configuring the interfaces together or separately does not apply to the Viewer Info page—the Viewer Info page contains metadata about the viewer, not viewer settings.

## Switch how you Configure the Interfaces

By default, the interfaces are configured together. In this case, there is only one tab, and its settings apply to all the interfaces.

### ▶ To switch to configuring the interfaces separately:

1. Click the **Configure Individually** hyperlink above the tab.  
A separate tab for each interface displays: Desktop, Tablet and Handheld. The hyperlink changes to say Configure Simultaneously.



If you are configuring the user interfaces individually, remember to click Apply Changes or Save Site before leaving a tab—otherwise, you will lose your changes.

2. To return to configuring the interfaces together, click **Configure Simultaneously**.  
You are prompted to select the interface whose configuration you want to keep.



Any changes that you made to the other two configurations will be lost.

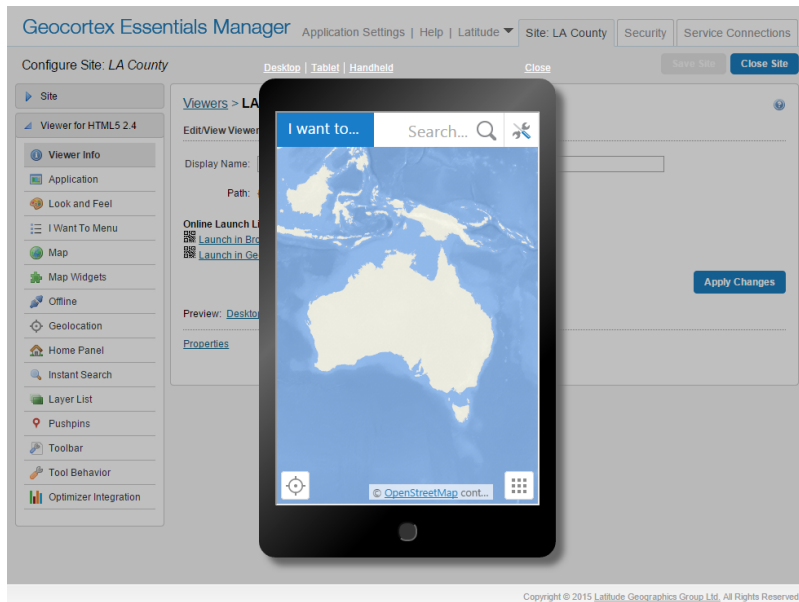
3. Select the interface whose configuration you want to keep from the drop-down list.
4. Click **OK**.  
The tabs are combined into one tab, and the configurations for this page's settings are made the same.

### 13.4.2 About the Live Preview

The Live Preview runs your viewer within a simulated desktop, tablet, or handheld device. You must click **Apply Changes** to run the current configuration, but you do not need to save the site. This enables you to test your configuration before you save the site.

You can use the viewer the same way in the Live Preview as in a browser—navigate the map, use menus and tools, search, run workflows, and so on.

To open the Live Preview, click **Apply Changes**, and then click one of the **Preview** links at the bottom of any HTML5 Viewer page in Manager. Use the links at the top of the Live Preview to switch between device types.



#### Live Preview for the Handheld interface

### 13.4.3 About User Interface Text

The HTML5 Viewer's user interface text is stored in language files, one for each library. Each text item in a language file is assigned to a placeholder called a "key". HTML5 viewers reference the keys, not the text. When the viewer runs in a browser, the viewer looks up the key in the appropriate language file and displays the text assigned to that key.

When you configure user interface text in Geocortex Essentials Manager, there are two ways you can specify the text. The method you use depends on how many languages the viewer will be available in.

- **Literal Text:** If your viewer is only going to be available in one language, type the text you want to use directly into Manager.
- **Text Key:** If you are going to make your viewer available in more than one language, type the key you want to use, or select it from the drop-down list that opens when you start typing. You must also create a language file for each language you are supporting. See [Translate Language Files on page 410](#).



Type @ to see the complete list of keys in the drop-down list—all text keys begin with @.

For more information on translating the HTML5 viewer, see [About Translating UI Text on page 410](#).

## 14 Configure a Viewer in Manager

### 14.1 Add a Viewer to a Site



In order to add a viewer using Manager, you must first install the viewer. See [Install the Viewer Framework Using a Viewer Template on page 10](#) for information.



#### To add a viewer to your site:

1. In Manager, edit the site that you want to configure, and then click **Viewers** in the side panel.
2. Click **Add Viewer**.  
The Create New Viewer dialog box opens.



If the Add Viewer icon is not available, click **Save Site**.

3. In the **Display Name** text box, type a display name for the viewer.

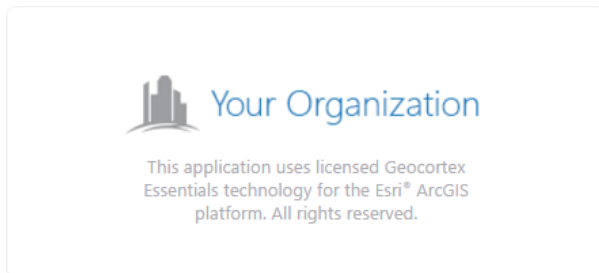


Avoid long viewer display names. Display names affect Windows file paths, which must be less than 260 characters.

4. In the **Template** drop-down list, select the template to base the viewer on.  
If there are no options in the Template drop-down list, there are no templates installed. See [Install the Viewer Framework Using a Viewer Template on page 10](#) for instructions on installing the HTML5 Viewer template.
5. Click **OK**.  
The dialog box closes and the viewer is added to your site.

### 14.2 Configure the Splash Screen

When a user launches an HTML5 viewer, the viewer displays a splash screen while the viewer loads. The splash screen contains an image and some licensing text. The default image contains a small graphic of a city and the words **Your Organization**.



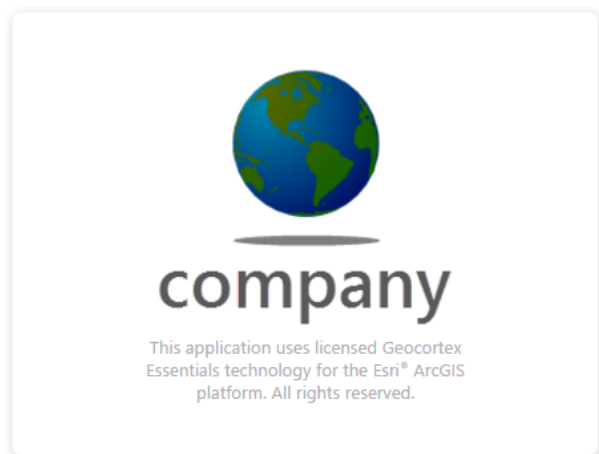
### Default splash screen

The default splash screen image is a placeholder that you can replace with a custom image. The splash screen has a built-in margin around the image, so the image does not touch the border or the licensing text. You cannot change the licensing text.

The default image is a 260 x 60 pixel PNG image of about 1.3 KB. To ensure reasonable performance, we recommend that you limit your image to about 4 KB, with dimensions no larger than 300 x 300 pixels. In the Handheld shell, if the image is larger than 260 x 260 pixels, the viewer resizes the image at run time so its dimensions are at most 260 x 260 pixels. This ensures that the image fits the smaller display area that handheld devices offer.



Animated images do not perform well in the splash screen, because the splash screen displays while the viewer is busy loading.



### Splash screen with a custom image

Manager's **Site Info** page has a **Splash Screen Image URL** setting where you configure the location of the image that the site's HTML5 viewers will use in the splash screen.



Before HTML5 Viewer 2.6, the splash screen image was configured by editing the `Index.html` file. We do not recommend this method, because changes to the `Index.html` file can be lost when you upgrade the Viewer.

The splash screen image can be stored anywhere, as long as the viewer can access the location. If you own the image, you can store it in a virtual directory with your other static content. In this case, you can use a relative URL when you configure the image's location. Alternatively, you can specify an absolute location, for example, a fully qualified URL to an image on the web.

---

► **To configure a custom splash screen image for a site's HTML5 viewers:**



If you configure a short URL in the viewer's `ViewerSettings.json.js` file, then you must configure the splash screen image in the `ViewerSettings.json.js` file with the short URL, instead of in Manager. For instructions, see [Short Viewer URLs](#).

1. If you want to create your own image to use in the splash screen, create it now.  
We recommend that you limit the image to about 4 KB in size, with dimensions no larger than 300 x 300 pixels. Using a transparent background usually gives the best results. The HTML5 Viewer supports any image file format that web browsers support. We recommend PNG or JPG.
2. If you want to store the image locally, save the image to a virtual directory.  
If you are going to use the same splash screen image for more than one site, put the image in the Global virtual directory. Otherwise put the image in the site's virtual directory.
3. In Essentials, edit the site that you want to configure, and then select **Site Info** from the side panel.
4. Enter the image's URL in the **Splash Screen Image URL** box using one of the following methods:
  - **Browse a Virtual Directory:** If the image is in a virtual directory, click **Browse** and select the image file.
  - **Enter the Fully Qualified URL:** If the image is not located in a virtual directory, type or paste the image's full URL, for example:  

```
http://host.domain.com/resources/images/myimage.png
```
5. Click **Apply Details**.
6. Click **Save Site**.

## 14.3 Edit a Viewer

► **To edit a viewer:**


1. Launch Manager.  
You can launch Essentials from your Start Menu. In the applications list, select the Latitude Geographics folder and the Geocortex Essentials Manager item inside.

2. Edit the site whose viewer you want to configure:
  - a. Go to the **Sites** tab.
  - b. If a different site is open for editing, click **Save Site**, and then click **Close Site**.
  - c. Follow the instructions for the view of the Site List that you are using:

- **Tiled View:**



World site in Tiled View


- Click the thumbnail image of the site that you want to edit, or
- Click the site's name, or
- Position the pointer over the thumbnail image of the site that you want to edit, and then use the  **Edit** icon.

- **List View:**



World site in List View

- Select the site's name or the  **Edit** icon .

3. Click **Viewers** in the side panel.
4. Click the **Edit** icon  beside the viewer that you want to configure.  
The viewer opens for editing.

## 14.4 Change Viewer Information

The Viewer Info page in Manager has high-level settings and meta-information about the viewer, such as the viewer's display name, the location of its configuration file, and a link to launch the viewer.

 **To open an HTML5 viewer's Viewer Info page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Viewer Info**.  
The Viewer Info page opens.



## Settings

The Viewer Info page has the following settings and information:

- **Display Name:** Used to refer to the viewer in Manager and in the REST API Sites Directory. The display name is not visible to the end user. However, the viewer ID is generated by the display name when the viewer is created and can appear in the viewer's URL.

Changing the display name after the viewer has been created does not change the ID.



Avoid long viewer display names. Display names affect Windows file paths, which must be less than 260 characters.

- **Global Unique ID:** A globally unique identifier for the viewer. It allows users to launch the viewer with a short launch URL. The value of the Global Unique ID is used for the `viewer` URL parameter. For example, if the value is `MyViewer`, you can launch the viewer with the following URL:

```
http://MyServer.com/Html5Viewer/index.html?viewer=MyViewer
```

The Global Unique ID cannot contain special characters such as `<`, `>`, `%`, `:`, `&`, `*`, `\`, `/`, and `?`.

For more information, see [Short Viewer URLs](#).



If you do not set the **Global Unique ID**, all viewers use the default Global Unique ID, which consists of the Site ID and the Viewer ID separated by a period character. For example,

```
http://MyServer.com/Html5Viewer/index.html?viewer=MySiteID.MyViewerID
```

- **Path:** The path to the viewer's configuration. `{SitePath}` represents the path to the folder containing the site's configuration file.
- **URL:** If you have deployed the viewer template to IIS multiple times, you can select which deployment you want the launch links and QR codes to use. This option is only visible if the template is deployed more than once.
- **Publish to ArcGIS Online:** Publishes the selected viewer URL as an item in ArcGIS Online. This only works if you used an ArcGIS Online identity to sign into Manager, and the selected viewer URL is not already published. If the item is already published, a link to the item on ArcGIS Online is displayed. The item can then be shared and configured in ArcGIS Online. A thumbnail is added in ArcGIS Online if the Essentials REST API is publicly accessible.

You can remove the item using the  Remove button next to the link.




Items can be orphaned if the viewer URL changes or the viewer is removed.

- **Publish to Portal:** Publishes the selected viewer URL as an item in Portal. This only works if you use a Portal for ArcGIS identity to sign into Manager, and the selected viewer URL is not already published. If the item is already published, a link to the item in Portal is displayed. The item can then be shared and configured in Portal. A thumbnail is added if Portal can access the Essentials REST API URL where the thumbnail exists.

You can remove the item using the  Remove button next to the link.

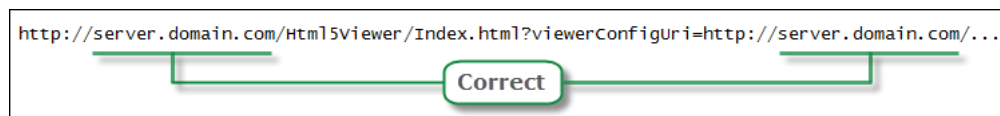


Items can be orphaned if the viewer URL changes or the viewer is removed.

- **Launch Links:** Hyperlinks and Quick Response (QR) codes (  ) that launch the viewer. Use the QR codes to launch the viewer in a device with a scanner, such as a smartphone. Hover the mouse pointer over a QR code to enlarge the code for scanning.
  - **Launch in Browser:** Launches the viewer in a browser. The Launch in Browser link uses the `configBase` URL parameter, which automatically detects the type of device that the viewer is running on—desktop computer, tablet, or handheld device—and uses the appropriate configuration for that device type. For more information on `configBase`, see [URLs for Secured Sites on page 54](#)
  - **Launch in Geocortex Mobile App Framework:** Launches the viewer in the Geocortex Mobile App Framework, provided you have purchased and installed the App.

If you did not deploy the viewer to IIS when you installed the viewer's template, the launch links and QR codes will be missing. See [Install the Viewer Framework Using a Viewer Template on page 10](#) for instructions on using the Post Installer to deploy the viewer to IIS.

If the viewer is blank when you launch it, and the [Live Previews](#) are also blank, edit the URL in the address bar to make all the references to the server (including the port) identical. We recommend fully qualified URLs. To correct this permanently, upgrade to Geocortex Essentials 3.11.1 or newer and redeploy the viewer.



#### Recommended form for an HTML5 viewer URL

- **Preview:** Hyperlinks that launch the viewer within an image of the specified device type, so you can see what the viewer looks like in its target device type. You can preview anything that you can do in a live viewer.
- **Properties:** Allows you to create a custom property for the viewer.

#### ► Before you leave the Viewer Info page:

1. Click **Apply Changes**.
2. To save your changes, click **Save Site**.

## 14.5 Configure Application-Wide Settings

The Application page in Manager has settings that apply to the viewer as a whole, such as the proxy page and geometry service to use.

You can configure additional application-wide settings in the configuration files. In particular, you can configure **datum transformations** to apply whenever the viewer performs an operation that requires projection, such as geocoding, measurement, or buffering. For information, see [Application-Wide Settings on page 58](#).

#### ► To open an HTML5 viewer's Application page in Manager:

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Application**.

The Application page opens.

► **Before you configure settings on the Application page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can always configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Application page to be different for one or more of the interfaces, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Application page has the following settings:

- **Proxy URI:** The URI to the proxy page, if you are using one. By default, the proxy URI is set to the ASP.NET proxy page that ships with the HTML5 Viewer, `proxy.ashx?`. The default URI is terminated with a question mark because the viewer attaches parameters to the URI. The question mark is optional—the viewer will add the question mark if necessary.

If you do not want to use a proxy page, remove `proxy.ashx?` from the **Proxy URI** box.

For instructions on setting up a proxy page, see [Set Up a Proxy Page on page 31](#).

- **Allow Unsafe Content:** If this checkbox is selected, content from a KML or GeoRSS layer that contains HTML markup is interpreted by the viewer. If you want the viewer to display the actual markup rather than what it represents, clear the checkbox. By default, unsafe content is not allowed.

If you have not added any KML or GeoRSS layers to your site, this setting has no effect.

- **Geometry Service URI:** The URI of the geometry service to use for projection and other geometry operations. The HTML5 Viewer's Geolocation feature uses the geometry service specified here to project the user's location from latitude/longitude coordinates to map coordinates.



As of version 2.5, the Geometry Service URI property of the HTML5 Viewer has been deprecated. To configure a geometry service, edit the site instead. If configured, the geometry service in the site overrides the one configured here. For more information, see "Geometry Services" in the *Geocortex Essentials Administrator Guide*.

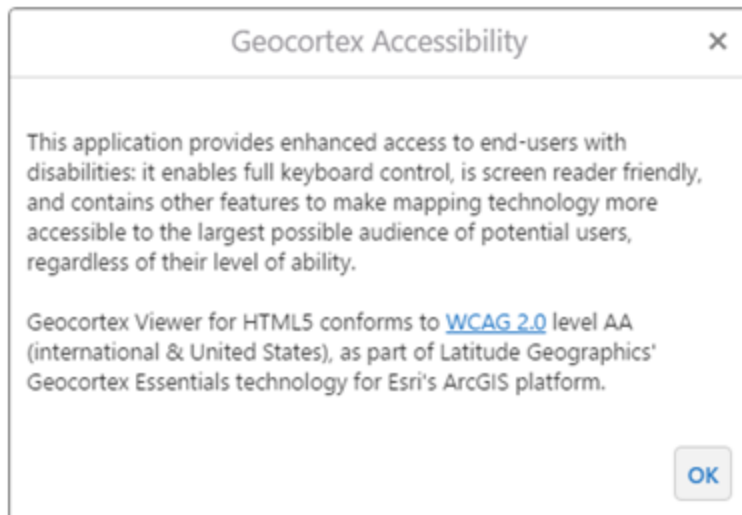
► **Before you leave the Application page:**

1. Click **Apply Changes**.
2. To save your changes, click **Save Site**.

## 14.6 Configure Accessibility

The Accessibility Panel informs users of the accessibility features in the HTML5 Viewer. To open the Accessibility Panel in the Desktop or Tablet interfaces, the user presses the  **Accessibility Button** in the viewer's banner. In the Handheld interface, the user can press the **View the accessibility panel** option in the [I Want To Menu](#).

You can configure the viewer to show or hide the Accessibility Button. You can also customize the title and content of the Accessibility Panel. The content of the Accessibility Panel is defined using HTML markup. The content can include formatted text, images, and links.



### The Accessibility Panel

The Accessibility Panel is implemented by the Accessibility Module. See [Accessibility Module on page 131](#) for information.

#### ► To open an HTML5 Viewer's Accessibility page in Manager:

1. In Manager, edit the viewer that you want to configure.
2. From the main menu, choose **Accessibility**.  
The Accessibility page opens.

#### ► Before you configure settings on the Accessibility page:


Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can always configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Home Panel page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Accessibility page has the following settings:

- **Include Enhanced Screen Reader Notifications:** When this checkbox is selected, map information is read out to users with screen readers. If you do not want map information to be read out to users with screen readers, clear the checkbox. By default, map information is read out to users with screen readers.
- **Include Accessibility Button:** In the Desktop and Tablet interfaces, when this checkbox is selected, the **Accessibility Button**  is available in the viewer. In the Handheld interface, when this checkbox is selected, the **View the accessibility panel** option is available in the [I Want To Menu](#). If you do not want the viewer to have an Accessibility option, clear the checkbox. By default, the Accessibility option is included.
- **Title:** The title that appears at the top of the Accessibility Panel.




If your viewer is going to be available in more than one language, enter the text key that the Accessibility Panel title is assigned to. For more information on using text keys, see [About User Interface Text on page 64](#).

The default title is `@language-accessibility-map-title`, which by default reads Geocortex Accessibility.

- **Content:** The HTML markup that defines the content of the Accessibility Panel. The Content box provides a Rich Text Editor with a toolbar at the top. By default, the Rich Text Editor is open in the Content box.

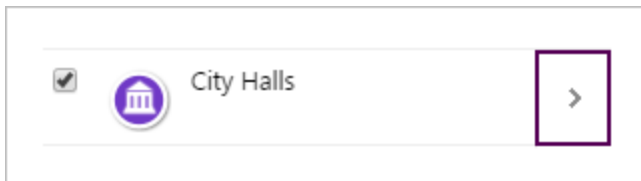


You can use a valid text key for the content as long as there are no spaces or HTML markup surrounding the text key. For more information on using text keys, see [About User Interface Text on page 64](#).

If the Rich Text Editor does not have a particular tool that you want, or you prefer to work directly in the HTML markup, use the  Show Source button at the end of the toolbar. This displays the HTML markup in the Content box, so you can edit the HTML markup directly.

To return to the Rich Text Editor, press Show Source  again.


- **Keyboard Focus Indicator Color:** Set the color for selected elements in the viewer with an RRGGBB hex code. For users who navigate the viewer using a keyboard, the focus indicator color outlines the selected HTML element so they can quickly find an interact with page elements. The default value is #550055.





A layer, City Halls, with the focus indicator color around the Actions menu button


- **Skip Links Menu:** The skip navigation function is supported in the Desktop shell. The items in the menu list define the links that open when initially tabbing in the Viewer. The first item in the list is the link that receives focus when the skip navigation links menu opens. Subsequent tabbing traverses each of the items in the menu. Users can quickly return to the skip navigation links menu by clicking the address field in the Viewer and then tabbing to open the menu. Users can also return to the initial item in the skip navigation links menu by

continuing to tab through the Viewer, or they can access the last link in the menu by tabbing back (SHIFT+TAB) through the Viewer.

To add a menu item, click **Add Menu Item**, update the fields, and click **OK**. You can see how to complete the fields by clicking the **Edit** icon  for an existing menu item.

To edit a menu item, click the **Edit** icon  for the item, update the fields in the dialog, and click **OK**.


To reorder the menu items, drag and drop the menu handle  to the new position in the list.

To remove a menu item, click the **Remove** icon  for the item.

Note that only 5 skip link items are supported in order to minimize the amount of tabbing required to access specific areas in the Viewer.

► **To configure the Accessibility options:**

1. On the **Accessibility** page, select the **Include Accessibility Button** checkbox.
2. In the **Title** box, type the title that you want to appear at the top of the Accessibility Panel. You can use a text key or the literal text.
3. In the **Content** box, create the content of the Accessibility Panel.

Use the Rich Text Editor that displays by default, or click the Show Source icon  to work in the HTML markup.

4. Click **Apply Changes**.
5. Click **Save Site**.

**See Also...**

[Accessibility on page 402](#)

[Accessibility Module on page 131](#)

[IWantToMenu Module on page 215](#)

[SkipLinks Module on page 359](#)

[About User Interface Text on page 64](#)

## 14.7 Change the Look and Feel

The Look and Feel page in Manager has settings that control the appearance of various aspects of the viewer, specifically:

- [Browser Title](#)
- [Banner](#)
- [Highlights](#)
- [Highlight Modes](#)
- [Data Frame](#)
- [Bottom Panel](#)
- [Results List](#)
- [Map Tips](#)

- [Feature Details](#)
- [Results Views](#)

▶ **To open an HTML5 viewer's Look and Feel page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Look and Feel**.  
The Look and Feel page opens.

▶ **Before you configure settings on the Look and Feel page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Look and Feel page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Look and Feel page has the following settings:

### Browser:

- **Title:** The title to display in the browser's title bar or tab.



If your viewer is going to be available in more than one language, enter the text key that the browser's title is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

The default text key is `@language-browser-title`. The text key's current value is shown below the Title box.

### Banner:

In the Desktop and Tablet interfaces, the banner is the area at the top of the viewer that optionally contains the title and logo. The Handheld interface does not have a banner.

- **Show Banner:** If this checkbox is selected, the banner shows in the viewer. If the checkbox is cleared, the viewer does not have a banner. By default, the viewer has a banner.
- **Title:** The text of the title to display in the banner. If you want control over the font and size, leave the Title box blank, create an image that contains the title, subtitle, and logo, and specify the image using one of the Image settings. You can use a text key or the literal text.
- **Title Color:** A valid HTML color to use for the title of the banner. For example, **red** or **#FF0000**.
- **Sub-Title:** The subtitle to display in the banner. The subtitle is the smaller text that appears under the title. The subtitle is optional. You can use a text key or the literal text.

If you want control over the font and size, leave the Sub-Title box blank, create an image that contains the title, subtitle, and logo, and specify the image using one of the Image settings.

- **Sub-Title Color:** A valid HTML color to use for the subtitle of the banner. For example, **green** or **#00FF00**.
- **Background Color:** A valid HTML color to use for the background of the banner, for example, **blue** or **#0000FF**.
- **Background Image:** The URL to the banner's background image, if you want a background image. Use this setting to add texture to the background color. Click **Browse** to browse to the image file.
- **Left Image:** The URL to the image that forms the left side of the banner. This setting is often used for the image containing the logo, title, and sub-title. Click **Browse** to browse to the image file. The maximum image height and width is 60px by 500px.
- **Left Image Description:** The alternative (**alt**) text for the Left Image. The alt text is used by screen readers. It is also used if the image cannot be displayed.
- **Right Image:** The URL to the image that forms the right side of the banner. Click **Browse** to browse to the image file.
- **Right Image Description:** The alternative (**alt**) text for the Right Image. The alt text is used by screen readers. It is also used if the image cannot be displayed.
- **Height:** The height of the banner, in pixels. The default banner is 60 pixels high.

#### Highlights:

Change the color of features that have been selected on the map. For example, if a user selected or identified a feature on the map.

- **Fill:** The fill color specified for highlighting, represented as an RGBA color. The default color is pale yellow defined as "RGBA(236, 236, 58, 0.1)". The first set of digits (236) is for the red color value, the second set (236) is for the green color value, the third set (58) is for the blue color value, and the last value (0.1) is for opacity, where 0.0 is fully transparent, and 1.0 is fully opaque.
- **Border:** The border color specified for highlighting. The default color is yellow, defined as "RGBA(255, 255, 151, 1)".
- **Outer Border:** The outer border color specified for highlighting. The default color is dark yellow, defined as "RGBA(200, 200, 0, 1)".
- **Focused Fill:** The fill color specified when the highlighted feature has focus. The default color is pale cyan, defined as "RGBA(0, 255, 255, 0.2)".
- **Focused Border:** The border color specified when the highlighted feature has focus. The default color is cyan, defined as "RGBA(0, 255, 255, 1)".
- **Outer Focused Border:** The outer border color specified when the highlighted feature has focus. The default color is dark cyan, defined as "RGBA(87, 170, 255, 1)".
- **Border width:** The border width specified for highlighting. The default value is 2 pixels.
- **Outer Border Width:** The outer border width specified for highlighting. The default value is 5 pixels.
- **Highlight Line Opacity:** The opacity setting for a highlighted line feature. The default is 0.5 pixels

#### Highlight Modes:



- **Mode:** Used to indicate how highlighting of features on the map is related to features in the Results List.
  - **Highlight Features On Page:** Indicates that only the features on the current page of the Results List are highlighted on the map.
  - **Highlight All Features:** Indicates that all of the features on all of the pages in the Results List are highlighted on the map.
  - **Do Not Highlight Any Features:** Indicates the none of the features on the pages in the Results List are highlighted on the map.

#### Data Frame:

Only the Desktop and Tablet interfaces have a Data Frame—the Handheld interface does not have a Data Frame.

- **Open By Default:** When this checkbox is selected, the Data Frame opens when the user launches the viewer. If you do not want the Data Frame to be open initially, clear the checkbox. By default, the Data Frame is closed. If you want the [Home Panel](#) to be visible when the user launches the viewer in the Desktop or Tablet interface, you must check the **Open By Default** checkbox.
- **Default Width:** The width of the Data Frame in pixels. The default width is **350** pixels. Alternatively, you can specify the amount as a percentage, for example, **25%**.



Specifying the amount as a percentage string ensures the panel size remains in proportion to the browser window until the user resizes the panel.

- **Minimum Width:** The minimum width to which the Data Frame can be resized. The default width is **200** pixels. Alternatively, you can specify the amount as a percentage, for example, **10%**.
- **Maximum Width:** The maximum width to which the Data Frame can be resized. The default width is **500** pixels. Alternatively, you can specify the amount as a percentage, for example, **50%**.

#### Bottom Panel:

- **Default Height:** (Desktop and Tablet interfaces only) The default height of the Bottom Panel. The default height is **400** pixels. Alternatively, you can specify the amount as a percentage, for example, **25%**.



Specifying the amount as a percentage string ensures the panel size remains in proportion to the browser window until the user resizes the panel.



To specify the percentage height of the Bottom Panel in the Handheld interface, manually edit the `bottomPanelHeightPercent` property in the Handheld configuration file. For more information, see [Shells Module on page 350](#).

#### Results List:

- **Content Field:** Specifies which setting to display for each feature in the Results List—the **Feature Description** or the **Feature Long Description**. The default is **Feature Long Description**.

**Map Tips:**

- **Content Field:** Specifies which setting to display as the main content of map tips—the **Feature Description** or the **Feature Long Description**. The default is **Feature Long Description**.

**Feature Details:**

- **Show Description:** The kind of description to display when viewing feature details, if any. Possible values include: **No Feature Description**, **Feature Description** and **Feature Long Description**. The default is **No Feature Description**.

**Results Views:**

- **Default View Type:** Specifies how results appear by default in the viewer. Possible values include **None**, **Results List**, and **Results Table**. For information about results views, see "Results Module" in the *Geocortex Viewer for HTML5 Administrator and Developer Guide*.

**Before you leave the Look and Feel page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

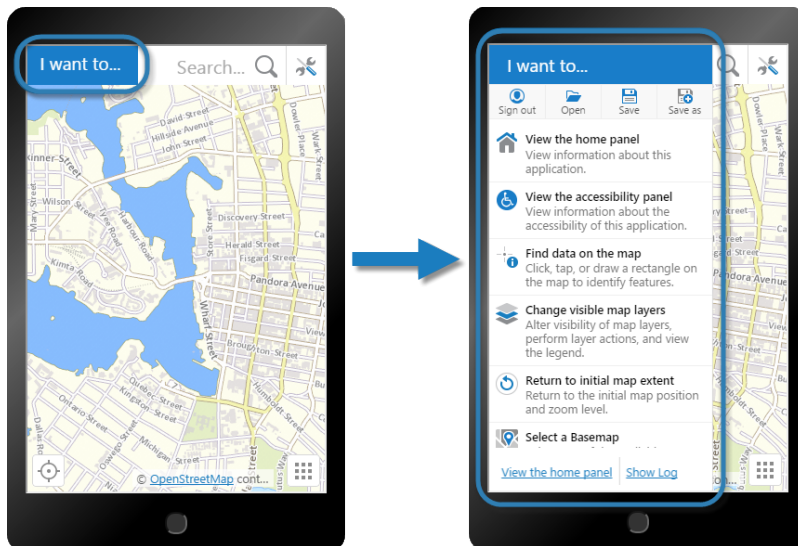
**See Also...**

[About User Interface Text on page 64](#)

[Configure the Home Panel on page 99](#)

## 14.8 Configure the I Want To Menu

The HTML5 Viewer's default configuration has a menu for common tasks, such as opening the list of map layers and returning to the initial extent. The menu is called the I Want To menu because the default text on the button that opens the menu is **I want to....** In the Desktop and Tablet interfaces, the I Want To button is in the top left corner of the map. In the Handheld interface, the button is at the top left of the screen.



### I Want To menu shown in the Handheld preview

Each menu item is represented by some text and optionally a description and small image. You can modify or remove these items, create new items, and change the order that the items appear in the menu. A menu item can run any Geocortex Viewer for HTML5 command. For a list of commands, see the [Geocortex SDK for HTML5 API Reference](#).

#### ► To open an HTML5 viewer's I Want To Menu page in Manager:

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **I Want To Menu**.  
The I Want To Menu page opens.

#### ► Before you configure settings on the I Want To Menu page:

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

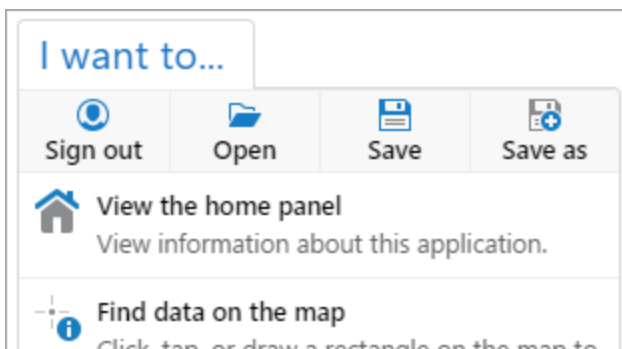
- If you are not sure, keep the settings the same by configuring the interfaces together—you can always configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the I Want To Menu page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Menu Configuration Settings

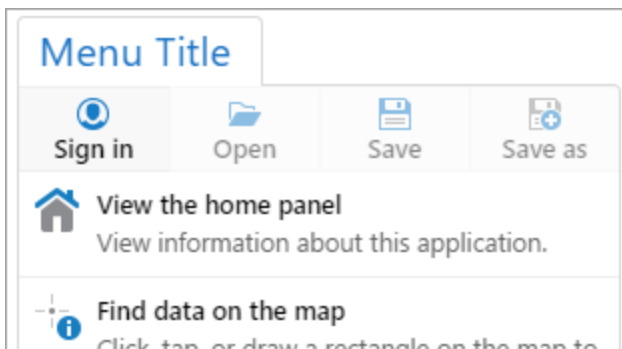
The I Want To Menu page has the following settings:

- **Show Menu:** If this checkbox is selected, the I Want To menu appears in the viewer.
- **Show Global Menu:** If this checkbox is selected, the Global Menu appears in the I Want To menu.  
The Global Menu allows users to sign into an account and save or load projects. For more information see [GlobalMenu Module on page 187](#).



The I Want To menu with the Global Menu enabled and active



- **Title:** The name of the I Want To menu.  
The default value is `@language-menu-title`, which populates the menu title with the default I Want To menu title for each included language. See [About User Interface Text on page 64](#) for more information on using text keys. Alternatively, you can manually enter another menu title.




The I Want To menu with the value "Menu Title" enter in the Title field


- **Primary Color of Button:** The color of the I Want To menu button as a RRGGBB hex code. The default value is #1A72C4.

## Menu Item Configuration Settings

You can add new menu items to the I Want To menu by selecting the  **Add Menu Item** button. You can edit existing menu items by selecting the  **Edit** button next to them. The following configuration settings are available for menu items:

- **Text:** The menu item's text.
- **Description:** (optional) A short description of what the menu item does.
- **Command:** The name of the command to run when the user selects the menu item. Click in the **Command** box to open a drop-down list of commands.
- **Command Parameter:** The parameter value to pass to the command when it runs. Parameters may either be simple strings or complex objects containing any number of parameters.
- **Icon URI:** (optional) The URI of the image to display next to the menu item. The image should be the size that you want it in the viewer. Valid file formats are PNG, BMP, JPG, and JPEG. The recommended image size is 24px by 24px.
- **Hide When Disabled:** If a menu item's functionality is unavailable or has been disabled, you can select this checkbox to hide the menu item in the viewer entirely. When this checkbox is not selected, the menu item is grayed out but still appears to users.

If you want to remove a menu item, use the  **Remove** button next to the item.

You can reorder existing menu items by clicking and dragging the  Sort handle next to each menu item.

▶ **Before you leave the I Want To Menu page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

**See Also...**

[About User Interface Text on page 64](#)

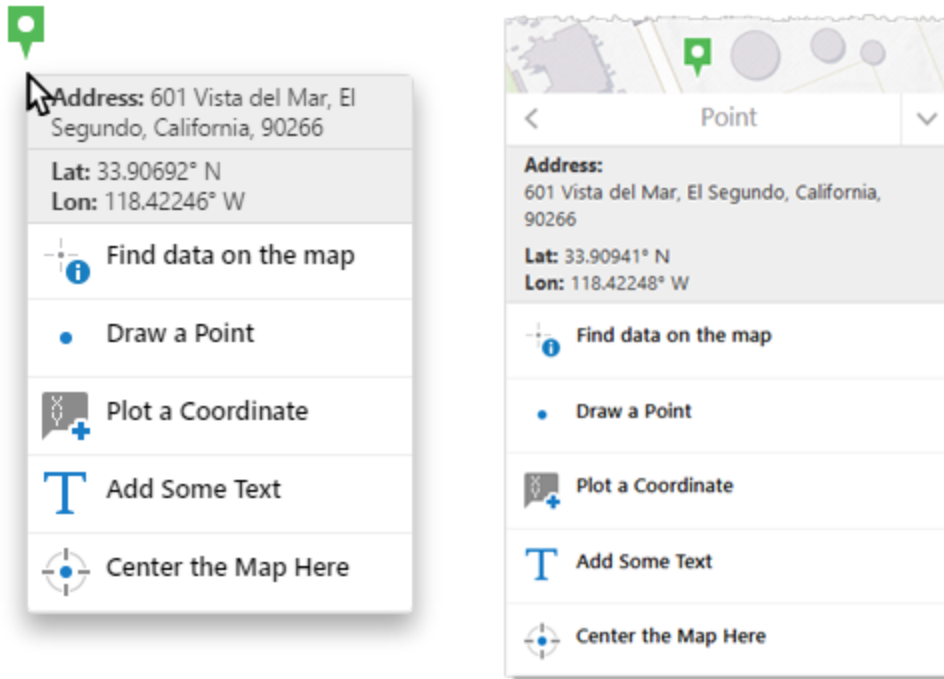
[Geocortex SDK for HTML5 API Reference on page 422](#)

## 14.9 Configure the Context Menus

You can configure context-sensitive menus throughout the viewer interface. Context menus allow users to perform common tasks that relate to the selected viewer element. For example, right-clicking or long-pressing on the map activates the map context menu. The map context menu displays tasks that a user might want to perform at a location on the map, like plotting a coordinate.

▶ **To open an HTML5 viewer's Look and Feel page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. From the menu, select **Context Menus**.



The map context menu in a desktop (left) and handheld interface (right)

## Types of Context Menus

Context menus can appear in a number of circumstances. For example, users can access context menus when they


- right-click or long-press on the map or map objects
- open a ☰ Panel Actions menu
- use an > Actions button
- open a map tip

For a complete list of configurable context menus see [List of Context Menus](#).

## List of Context Menus

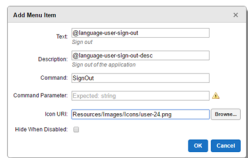
The following context menus are configurable. Users can access each context menu in the following way:

Menu	Description
Coordinate Actions	Select the ⋮ More button next to any plotted coordinate when Plot Coordinates is the active panel.
Coordinates List Actions	Open the ☰ Panel Actions menu when Plot Coordinates is the active panel.



Menu	Description
Feature Actions	Open the ☰ Panel Actions menu when a feature is active in the navigation panel.
Global Menu	<p>Open the I Want To Menu. The Global Menu appears as four icons across the top of the open menu. The recommended image size is 24px by 24px.</p> <p>The default menu items included are for Sign In or Sign Out, Open Project, Save Project, and Save Project As.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p> The Global Menu is designed to hold a maximum of four items. Users are discouraged from configuring the menu in a manner that allows for more than four items to be visible at the same time.</p> </div>
Layer Actions	Open the > Actions menu next to a layer when Layers is the active panel.
Layer List Actions	Open the ☰ Panel Actions menu when Layers is the active panel.
Legend Actions	<p>Open the ☰ Panel Actions menu when Legend is the active panel.</p> <p>To show the Legend panel, open the Layers panel, open the ☰ Panel Actions menu and select Show Legend. For more information, see <a href="#">Legend Module on page 244</a>.</p>
Map Context Menu	<p>Right-click or long-press any point on the map.</p> <p>The Map Context Menu has additional features that you can enable or disable. For more information see <a href="#">Configure the Map Context Menu</a>.</p>
Map Service Actions	Open the > Actions menu next to a map service when Layers is the active panel.
Map Tip Actions	Appear as a list of links on the active map tip.
Project Actions	Open the > Actions menu next to a project when Projects is the active panel.
Projects List Actions	Open the ☰ Panel Actions menu when Projects is the active panel.
Results List Actions	Open the ☰ Panel Actions menu when the Results List is the active panel.
Results Table Actions	Open the ☰ Panel Actions menu when the Results Table is activated.
Selection Actions	<p>When the Results List is the active panel or the Results Table is activated, open the ☰ Panel Actions menu and select the Combine Results item. The Combine Results item activates the Selection Actions menus.</p> <p>See <a href="#">Selection Module on page 341</a> for more information about selections.</p>

## Add and Modify Menu Items

From each configurable context menu, use the  **Add Menu Item** button to open the Add Menu Item dialog and create a new menu item.




### The Add Menu Item dialog

Choose  **Edit** or  **Remove** to edit or remove next to the menu item you want to edit or remove.



After you have added or edited menu items, select **Apply Changes** and **Save Site** to save your changes to the viewer settings.

## Reorder Menu Items

You can rearrange any existing menu items use the  sort handle and drag each menu item up or down in the list of items.

## Contextual Menu Items

Some menu items do not always appear in a context menu. For example, if a layer is not editable, then a feature's Panel Actions menu would not show the Edit Feature menu item. In the same way, some menu items are contextual based on the user's permissions level. If a user is not logged into their Identity Server identity or their ArcGIS Online account, menu may not appear.

## Context Menu Item Configuration

The following settings can be set for menu items:

- **Text:** The menu item's text.
- **Description:** Optional. A short description of what the menu item does.
- **Command:** The name of the command to run when the user selects the menu item. Available commands are suggested if you have the command text box selected.
- **Command Parameter:** The parameter value to pass to the command when it runs. Parameters may either be simple strings or complex objects containing any number of parameters.  
The Command Parameter setting is grayed out for commands that cannot take a parameter.
- **Icon URI:** Optional. The URI of the image to display next to the menu item. The image should be the size that you want it in the viewer. Valid file formats are PNG, BMP, JPG, and JPEG.  
Use the **Browse...** button to choose an icon that exists in your site's virtual directory.
- **Hide When Disabled:** Select this checkbox to hide menu items if they are disabled.  
For example, if a user does not have permission to edit a feature, the Edit Feature action would be hidden from the user.





The Command and Command Parameter settings are not available when a menu item uses multiple commands via the `batch` property. In this case, those settings are hidden. You must edit batch commands directly in the `Desktop.json.js`, `Tablet.json.js`, and `Handheld.json.js` configuration files.

## 14.9.1 Configure the Map Context Menu

The Map Context Menu page has the following settings:

- **Enable Context Menu:** To display the Map Context Menu when the user right-clicks or long-presses a point on the map, select this checkbox; otherwise, clear this checkbox. This checkbox is selected by default.
- **Show Coordinates:** To display the coordinates of the point right-clicked or long-pressed, select this checkbox; otherwise, clear this checkbox. This checkbox is selected by default.
- **Show Reverse Geocoder:** To display the address of the point right-clicked or long-pressed, select this checkbox; otherwise, clear this checkbox. This checkbox is selected by default.

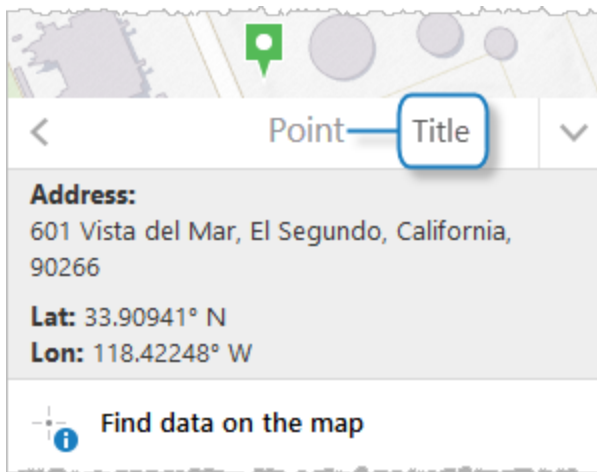


For this option to work, your site must have at least one Geocoder configured with the **Default Reverse Geocoder** setting selected. For more information, see "Geocoding Services" in the *Geocortex Essentials Administrator Guide*.

- **Show Menu:** To display the menu items of the Map Context Menu, select this checkbox; otherwise, clear this checkbox. This checkbox is selected by default.
- **Title:** If your viewer is going to be available in one language only, type the menu title. This title appears in the Bottom Panel of the Handheld interface.



If your viewer is going to be available in more than one language, enter the text key that the menu title is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.



Title for the Map Context Menu

For more information about context menu configuration, and adding and modifying menu items, see [Configure the Context Menus](#) on page 81.

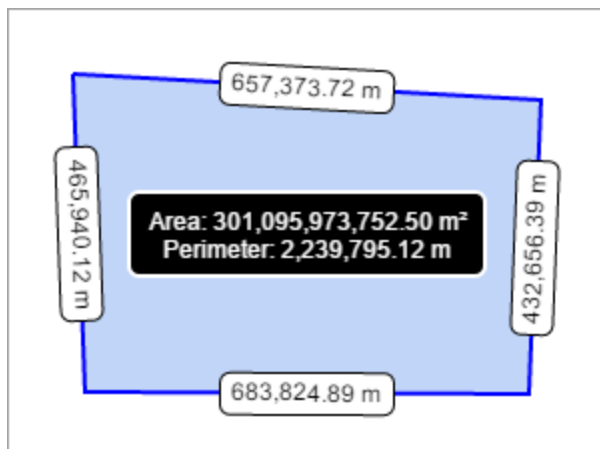
See Also...

[About User Interface Text](#) on page 64

[Geocortex SDK for HTML5 API Reference](#) on page 422

## 14.10 Configure Measurement

The viewer includes default configuration for displaying measurements in the viewer. Whenever users create drawings on the map, the measurement configuration is used to display the measurement lines and text. The Measurement page in the Management Pack allows you to modify the configuration used.



An example of a polygon's area and line measurement

### General Properties

You can configure the lines, fill, and text for all measured markup objects. The following properties can be configured:

- **Markup Layer Name:** Set the name of the markup layer. The default value is `Drawings`.



The markup layer does not appear in the layer list. The Markup Layer Name is how Essentials references drawings in the viewer. Generally, there is no reason to change this setting.

- **Line Color:** The line color for markup objects, set in RGB hex format. The default value is `#0000FF`.
- **Fill Color:** The fill color for markup objects, set in RGB hex format. The default value is `#6495ED`.
- **Text Color:** The color of the line measurement text, set in RGB hex format. The default value is `#000000`.
- **Highlight Color:** The color behind the text on line measurements, set in RGB hex format. The default value is `#FFFFFF`.
- **Outline Color:** The color of the border around the text on line measurements, set in RGB hex format. The default value is `#000000`.

- **Outline Width:** The width of the border around the highlighted text on line measurements, set in pixels. The default value is 1.
- **Highlight Radius:** The border radius of the highlighted line measurement text, set in pixels. The default value is 5.  
If the highlight radius is set to 0, the highlighted border is square. Higher values, like 5 or 10, determine how round the border is.
- **Text Size:** The size of the text used for line measurements, set in pixels. The default value is 12.
- **Add Markup to Map By Default:** Select this checkbox to add markup to the map by default. This means that when users finish drawing and move on to another activity in the viewer, their drawings remain on the map. This checkbox is checked by default.

## Total Measurement Properties

You can configure the measurement display for polygon drawings. Total measure, like the area of a polygon, is displayed on top of the drawing. The following total measurement properties can be configured:

- **Total Measurement Text Color:** The color of text for the area measurements, set in RGB hex format. The default value is #FFFFFF.
- **Total Measurement Highlight Color:** The color behind the text on area measurements, set in RGB hex format. The default value is #000000.
- **Total Measurement Outline Color:** The color of the border around area measurements, set in RGB hex format. The default value is #FFFFFF.
- **Total Measurement Outline Width:** The width of the border around area measurements, set in pixels. The default value is 2.

## 14.11 Configure the Map

The Viewer's Map page in Manager has settings that enable you to extend or restrict the scale range of the map.

### ▶ To open an HTML5 viewer's Map page in Manager:

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Map**.  
The Map page opens.

### ▶ Before you configure settings on the Map page:

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Map page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Map page has the following settings:

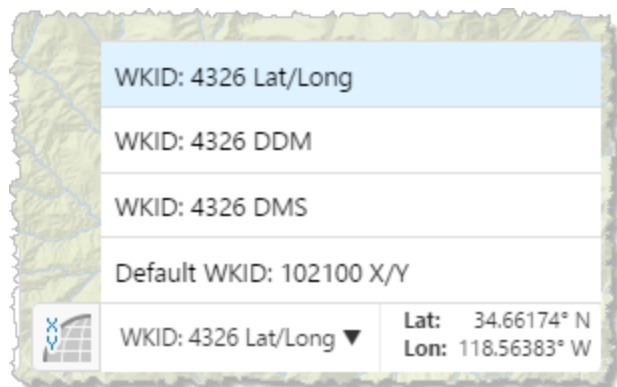
- Override Scale:** The Override Scale settings enable you to override the default scale range of the map. By default, the primary map service determines the scale range of the map. The map's Zoom control has the same range as the map's primary map service. Each level of detail in the primary map service is one zoom level in the Zoom control. Users cannot zoom in closer than the primary map service's minimum scale, or zoom out farther than the primary map service's maximum scale.

A site may contain a service, often a dynamic service, with a larger scale range than the primary map service. In this case, users will not be able to see the entire map service—they can only see the part that lies within the primary map service's scale range. The Override Scale settings enable you to extend the map's scale range so users can see the parts of the map service that lie outside the primary map service's scale range.

You can also use the Override Scale settings to restrict the map's scale range.

  - Override Minimum Scale:** The minimum scale at which the map is visible in viewers. For example, if the map's minimum scale is 1:5,000,000, you could extend the minimum scale to 1:10,000,000 in Essentials. To configure the minimum scale, enter the scale's denominator, without commas or other punctuation. For example, to set the minimum scale to 1:10,000,000, enter **10000000** in the **Override Minimum Scale** box.
  - Override Maximum Scale:** The maximum scale at which the map is visible in viewers. For example, if the map's maximum scale is 1:2,000, you could extend the maximum scale to 1:500 in Essentials. To configure the maximum scale, enter the scale's denominator, without commas or other punctuation. For example, to set the maximum scale to 1:500, enter **500** in the **Override Maximum Scale** box.
- Map Coordinate Systems:** The Map Coordinate Systems settings allow you to control how the map coordinates are displayed.

The map coordinates show the position of the mouse pointer. By default, the coordinates appear at the bottom left of the map, beside the scale bar:



Map coordinates with the coordinate system menu open

- **Default Coordinate Display Types:**
  - **Decimal Degrees (Lat/Long) - Default Geographic Coordinate System Spatial Reference:** To include the default geographic coordinate system displayed in latitude and longitude, select this checkbox; otherwise, clear this checkbox. By default, this checkbox is selected.
  - **Degrees Decimal Minutes (DDM) - Default Geographic Coordinate System Spatial Reference:** To include the default geographic coordinate system displayed in decimal degrees and minutes, select this checkbox; otherwise, clear this checkbox. By default, this checkbox is selected.
  - **Degrees Minutes Seconds (DMS) - Default Geographic Coordinate System Spatial Reference:** To include the default geographic coordinate system displayed in degrees, minutes and seconds, select this checkbox; otherwise, clear this checkbox. By default, this checkbox is selected.
  - **Projected Coordinates (X/Y) - Primary Map Spatial Reference:** To include the primary map's projected coordinate system displayed in X and Y, select this checkbox; otherwise, clear this checkbox. By default, this checkbox is selected.
- **Default Geographic Coordinate System WKID:** The well-known ID of the default coordinate system. The default is **4326**.



For a list of WKIDs, see Esri's [Geographic Coordinate Systems](#) or [Projected Coordinate Systems](#) documentation.

- **Coordinate Decimal Precision:** The number of decimal places to use for coordinates. The default is **5**.



This precision setting cannot exceed the maximum representable IEEE 754 floating point. This is specified in the IEEE 754 standard. See [double-precision floating-point format](#) and [significant figures](#) for more information.

- **Alternate Coordinate Systems:** A list of coordinate systems to make available as options in the coordinates widget. To add a coordinate systems, click **Add** and configure the settings:
  - **Display Name:** The name to display in the coordinates widget.
  - **WKID:** The well-known ID of the coordinate system. Cannot be set if **WKT** is already specified.



For a list of WKIDs, see Esri's [Geographic Coordinate Systems](#) or [Projected Coordinate Systems](#) documentation.

- **WKT:** The well-known text of the coordinate system. Cannot be set if **WKID** is already specified.



For a list of WKTs, see Esri's [Geographic Coordinate Systems](#) or [Projected Coordinate Systems](#) documentation.

- **Output Labels:** The units to use for the coordinates. Possible values include:

- **DDM:** Degrees, Decimal Minutes
- **DMS:** Degrees, Minutes, Seconds
- **Lat/Long:** Latitude, Longitude
- **X/Y:** X, Y (in the units of the coordinate system)
- **MGRS:** Military Grid Reference System, if configured
- **USNG:** United States National Grid, if configured



Note that:

- The MGRS and USNG coordinate systems require ArcGIS Server version 10.3 or later.
- MGRS and USNG are not supported in offline mode.

For example, to configure the MGRS coordinates:

- a. Click **Add** under the **Alternate Coordinate Systems** field to open the Add/Edit Coordinate System dialog.
- b. Type a name for display purposes in the **Display Name** field.
- c. Leave the **WKID** and **WKT** fields blank.
- d. Select **MGRS** in the **Output Type** list.
- e. Click **OK** to add MGRS as an alternate coordinate system.
- f. Click **Apply Changes** and then **Save Site**.
- g. Refresh the Viewer if it is open.  
**MGRS** is now available as a selection in the list of coordinate systems.

► **Before you leave the Map page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

## 14.12 Configure Map Widgets

The Map Widgets page in Manager has settings for:

- [Bookmarks](#)
- [Scale Bar](#)
- [Scale Input Box](#)
- [Overview Map](#)

- [Map Coordinates](#)
- [Time Slider](#)

▶ **To open an HTML5 viewer's Map Widgets page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Map Widgets**.  
The Map Widgets page opens.

▶ **Before you configure settings on the Map Widgets page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Map Widgets page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Bookmarks

- **Enabled:** When this checkbox is selected, the Bookmarks feature is enabled—you can add the Bookmarks map widget or Bookmarks tool, so users can open the Bookmarked Locations menu. The Bookmarked Locations menu allows users to jump to existing bookmarks and create new bookmarks to add to the menu.  
To show the Bookmarks map widget, select the **Show Bookmarks Button** checkbox. When the widget is clicked, the Bookmarked Locations menu opens beside the widget.  
To add the Bookmarks tool to the toolbar, follow the instructions to add a button in [Configure the Toolbar on page 108](#). When the Bookmarks tool is clicked, the Bookmarked Locations menu opens next to the Bookmarks map widget if the widget shows on the map. If the widget does not show on the map, the Bookmarked Locations menu opens in a modal window.  
By default, the Enabled checkbox is selected.
- **Show Bookmarks Button:** When this checkbox is selected, the Bookmarks map widget appears on the map, provided at least one bookmark is defined. When the widget is clicked, the Bookmarked Locations menu opens beside the widget.  
When Show Bookmarks Button is selected, the viewer hides the Bookmarks map widget if there are no bookmarks defined. In this case, users can use the **Bookmark current map extent I Want To** menu item to create bookmarks. As soon as a user has created one bookmark, the Bookmarks widget shows on the map.  
By default, the Show Bookmarks Button checkbox is cleared.



To select the **Show Bookmarks Button** checkbox, the **Enabled** checkbox must be selected.

## Scale Bar

The scale bar appears at the bottom left of the map.

- **Show Scale Bar:** To display the scale bar, select this checkbox; otherwise, clear this checkbox. It is selected by default.
- **Scale Bar Style:** Select a style for the scale bar from this drop-down menu. The possible choices are **Ruler** or **Line**. The default is **Ruler**.
- **Scale Bar Unit:** Select the units of measurement for the scale bar from this drop-down menu. The possible choices are **US Customary**, **Metric** or **Both (US Customary and Metric)**. The **Both (US Customary and Metric)** option can only be used with the **Line** scale bar style. The default is **Metric**.



Dual-unit Line scale bar (left) and metric Ruler scale bar (right)

- **Show Background:** To display a rectangular background for the scale bar, select the Show Background checkbox. The background prevents other widgets from overlapping the scale bar. In the screen capture above, both scale bars have a background. If you do not want the scale bar to have a background, clear the checkbox. It is selected by default.

## Scale Input Box

The scale input box allows users to choose the scale of the map.



### The scale input box

If it is enabled, the scale input box can be toggled by the scale input button at the bottom left of the map.



### The scale input button

- **Show Scale Input Box:** To show the scale input button that toggles the scale input box, select this checkbox.
- **Open By Default:** To show the expanded scale input box by default, select this checkbox.

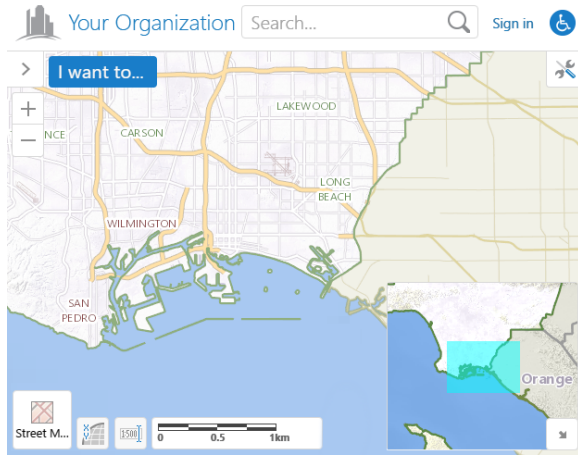


The scale input box is visible if both Show Scale Input Box and Open By Default checkboxes are selected. For example, if you select the Open By Default checkbox but leave the Show Scalebar Input Box deselected, the scale input box is hidden.



## Overview Map

The overview map appears at the bottom right of the map.

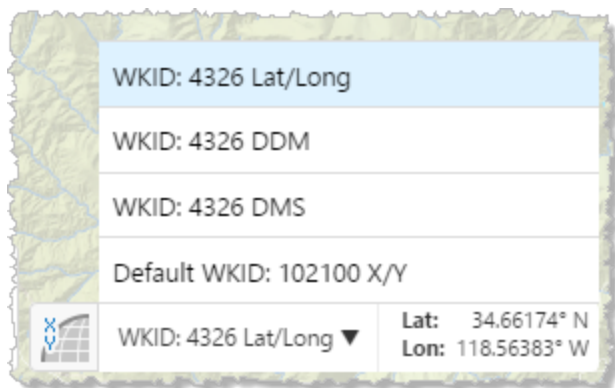


### Example of an overview map

- **Show Overview Map:** To enable the overview map, select this checkbox; otherwise, clear this checkbox. It is selected by default.
- **Open By Default:** To open the overview map when the viewer starts, select this checkbox; otherwise, clear this checkbox. It is cleared by default.
- **Extent Scale Factor:** The scale factor to use for the overview map in relation to the current map extent. The default is **2**.
- **Color of Visible Extent:** A valid HTML color to use for the rectangle that represents the current map extent. The default is **#00FFFF**.

## Map Coordinates

The map coordinates show the position of the mouse pointer. By default, the coordinates appear at the bottom left of the map, beside the scale bar:



Map coordinates with the coordinate system menu open

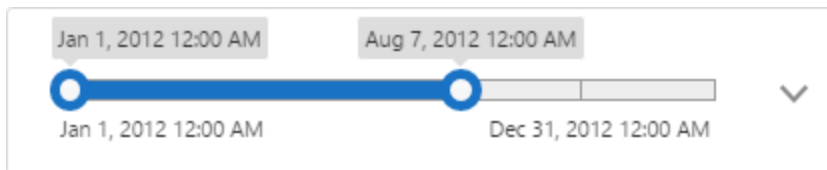
- **Show Map Coordinates:** To show the map coordinates widget on the map, select this checkbox. For the widget to display, at least one coordinate system must be specified—either Display Basemap's Coordinate System is enabled or at least one Alternate Coordinate System is configured, or both.  
If you do not want the coordinates widget to show, clear the Show Map Coordinates checkbox. The checkbox is selected by default.
- **Open By Default:** To open the map coordinates widget when the viewer starts, select this checkbox; otherwise, clear this checkbox. It is cleared by default.



The Map Coordinates widget is only available for the Desktop interface.

## Time Slider

The time slider map widget appears in the viewer when a time-aware layer has been configured in Manager. For more information, see [TimeSlider Module on page 372](#).



### The time slider map widget

- **Enable Time Slider:** Select this checkbox to enable the time slider map widget.
- **Animate Time Slider:** Select this checkbox to enable smoother time slider animations when playing a timeline or using the Step Back and Step Forward buttons.
- **Activate By Default:** Select this checkbox to activate the time slider map widget by default.

### ► Before you leave the Map Widgets page:

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

## 14.13 Configure a Viewer for Offline Use

Through the Geocortex Mobile App Framework, the HTML5 viewer supports access to feature layers and basemaps when there is no network connection. In offline mode, users can navigate the feature layers, perform searches and identify operations, and if the layers are editable, create, modify, and remove features. When the viewer is connected again, they can upload the changes to the server.

As of Geocortex Viewer for HTML5 2.0, launching a viewer in offline mode requires the Geocortex Mobile App Framework. For more information, see [Offline on page 401](#).

► **To open an HTML5 viewer's Offline page in Manager:**


1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Offline**.  
The Offline page opens.

## Settings

The Offline page has the following settings:



- **Add to List of Available Viewers:** Whether this viewer should be offered to be added in the Geocortex Mobile App Framework.
- **Automatically Download:** Whether this viewer should be automatically downloaded in the Geocortex Mobile App Framework.

► **To configure an HTML5 viewer for offline use:**

1. In Essentials Manager, edit the site that contains the viewer.
2. In the side panel, click **Viewers**, then click the **Edit** icon  beside the desired viewer.
3. In the **Viewer for HTML5** side panel, click **Offline**.
4. To offer this viewer to be added in the Geocortex Mobile App Framework, select the **Add to list of Available Viewers** checkbox.



If the site is secured, the viewer will only appear in the Available Viewers list if the user has permission to access the viewer. For information on setting permissions on viewers, see "Permissions" in the *Geocortex Essentials Administrator Guide*.

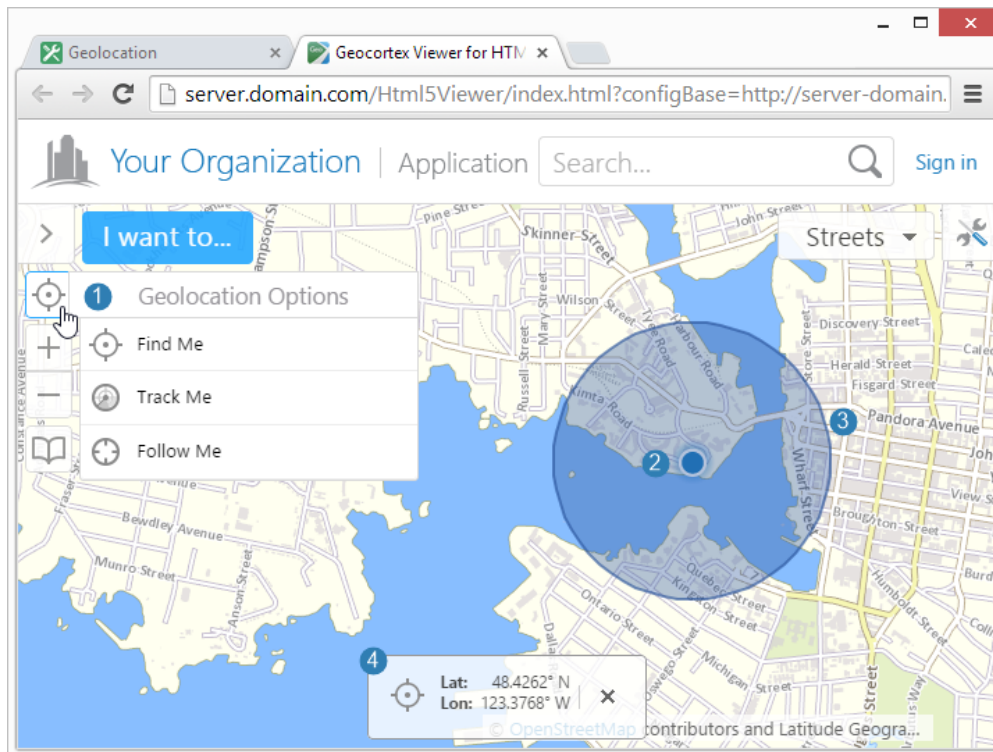
5. To automatically download this viewer in the Geocortex Mobile App Framework, select the **Automatically download** checkbox.
6. Click **Apply Changes**.
7. In the **Viewer for HTML5** side panel, click **Toolbar**.
8. In the **Configured Toolbar** section, ensure the **Offline Maps** tool  is present. If not, drag the **Offline Maps** tool  from the **Available Tools** section to the **Configured Toolbar** section.
9. Click **Apply Changes**.
10. Click **Save Site**.

## 14.14 Configure Geolocation

Geolocation locates the user on the map. The HTML5 Viewer supports the following geolocation options:

- **Find Me:** Pans the map to the user's location and marks the location with an indicator.
- **Track Me:** Tracks the user's location with an indicator, without panning the map.

- **Follow Me:** Follows the user's location with an indicator and pans the map as the user's location changes.



Geolocation menu (1), location indicator (2), accuracy circle (3), and geolocation coordinates (4)

## Browser Support

Web browsers return geolocation results in WGS 84. If your map is not in Web Mercator or WGS 84, you must configure an ArcGIS geometry service so the geolocation widget can project from latitude/longitude to the map's coordinates. For instructions on configuring a geometry service, see [Configure Application-Wide Settings on page 70](#).

Each web browser handles geolocation differently. For example, as of Chrome 50.0, your site needs to be secured with HTTPS in order to use the Geolocation API. If you or your users intend to use geolocation, ensure that your preferred web browser uses geolocation and that it is compatible with your site. Here are two common scenarios where geolocation is not available to users:

- **Chrome 50+:** Your site needs to be secured with HTTPS in order to use the Geolocation API.
- **Internet Explorer 8:** Geolocation is not supported in Internet Explorer 8 or older. If you use Internet Explorer Enterprise Mode, you may be emulating a version of Internet Explorer 8.



It is common for devices to ask permission before websites can use geolocation. If geolocation is not working, ensure that the viewer has been given sufficient geolocation privileges. Also ensure that the devices location services are enabled.

In order for an HTML5 viewer to perform geolocation operations, the user's device must have a built-in GPS (Global Positioning System), a Wi-Fi connection, or a wired connection. GPS provides the most accurate geolocation results. Wi-Fi is less accurate than GPS, and geolocation using a wired connection is less accurate than Wi-Fi.

Some devices use a mix of GPS and Wi-Fi to obtain a location. This can result in unpredictable readings. Accuracy is improved by setting the device to GPS-only mode, however, this drains the battery more quickly in some devices.

Geolocation is configured on the viewer's Geolocation page in Manager. You can configure additional geolocation settings in the viewer's configuration files. For information, see [Geolocate Module on page 184](#).

### ▶ To open an HTML5 viewer's Geolocation page in Manager:

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Geolocation**.  
The Geolocation page opens.

## Settings



By default, the Geolocation page is set to configure the Desktop, Tablet, and Handheld interfaces separately. For more information, see [About Configuring Multiple Interfaces on page 62](#).

The Geolocation page has the following settings:

### Geolocation:

- **Enable Single-Reading Geolocation:** Select this checkbox if you want to make single-reading geolocation (**Find Me**) available in the Geolocation Options menu; otherwise, clear this checkbox. The Find Me feature pans the map to the user's current location and marks the location with an indicator.  
By default, this setting is enabled for the Tablet and Handheld interfaces, and disabled for the Desktop interface.
- **Enable Geolocation Tracking:** Select this checkbox if you want to make tracking (**Track Me**) available in the Geolocation Options menu; otherwise, clear this checkbox. The Track me feature tracks the user's location with an indicator, without panning the map.  
By default, this setting is enabled for the Tablet and Handheld interfaces, and disabled for the Desktop interface.
- **Enable Geolocation Following:** Select this checkbox if you want to make following (**Follow Me**) available in the Geolocation Options menu; otherwise, clear this checkbox. The Follow Me feature follows the user's location with an indicator and pans the map as the user's location changes.  
By default, this setting is enabled for the Tablet and Handheld interfaces, and disabled for the Desktop interface.
- **Display Geolocation Accuracy Circle:** Select this checkbox if you want to display the geolocation accuracy circle; otherwise, clear this checkbox. It is checked by default. When checked, the indicator for the user's position includes a surrounding circle that indicates the margin of error of the user's position; the user's actual position should lie somewhere within this circle.
- **Geolocation Indicator Image:** The URL to the image that represents the geolocation indicator. Type the URL in the text box. Alternatively, click **Browse** to use an image on the server; if necessary, upload an image to the server by clicking **Upload** and selecting an image file to upload. The default is an image of a blue dot located at **Resources/Images/Icons/geolocate-position-32.png**.
- **Zoom Upon Geolocation:** Select this checkbox if you want to zoom to the user's location upon single-reading

geolocation (**Find Me**) or following (**Follow Me**); otherwise, clear this checkbox. It is checked by default.

- **Zoom Value:** The scale to which to zoom when **Zoom Upon Geolocation** is selected. The default is **50000** (to 1).

#### Geolocation Coordinates:

- **Display Coordinates for Single-Reading Geolocation:** Select this checkbox to display the coordinates of the user's location, in addition to showing the geolocation indicator, when the user performs a **Find Me** operation. If you do not want to show the coordinates for Find Me operations, clear the checkbox.
- **Display Coordinates for Geolocation Tracking:** Select this checkbox to display the coordinates of the user's location, in addition to showing the geolocation indicator, when the user performs a **Track Me** operation. If you do not want to show the coordinates for Track Me operations, clear the checkbox.
- **Display Coordinates for Geolocation Following:** Select this checkbox to display the coordinates of the user's location, in addition to showing the geolocation indicator, when the user performs a **Follow Me** operation. If you do not want to show the coordinates for Follow Me operations, clear the checkbox.
- **Coordinate Format:** Select the units to use for displaying the coordinates. The options are:
  - **DDM:** Degrees, Decimal Minutes
  - **DMS:** Degrees, Minutes, Seconds
  - **Lat/Long:** Latitude, Longitude
  - **X/Y:** X, Y (in the units of the coordinate system)
  - **MGRS:** Military Grid Reference System, if configured
  - **USNG:** United States National Grid, if configured

By default, X/Y coordinates are shown in the map's spatial reference. To change the spatial reference that is used for the coordinates, use the **WKID** setting to specify the well-known ID of the coordinate system that you want to use.



Note that:

- The MGRS and USNG coordinate systems require ArcGIS Server version 10.3 or later.
- MGRS and USNG are not supported in offline mode.

- **Coordinate Decimal Precision:** The number of decimal digits to show in the coordinates.



This precision setting cannot exceed the maximum representable IEEE 754 floating point. This is specified in the IEEE 754 standard. See [double-precision floating-point format](#) and [significant figures](#) for more information.

#### ▶ Before you leave the Geolocation page:

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.

See [About the Live Preview on page 64](#) for instructions.

- To save your changes, click **Save Site**.

See also...

[Geolocate Module on page 184](#)

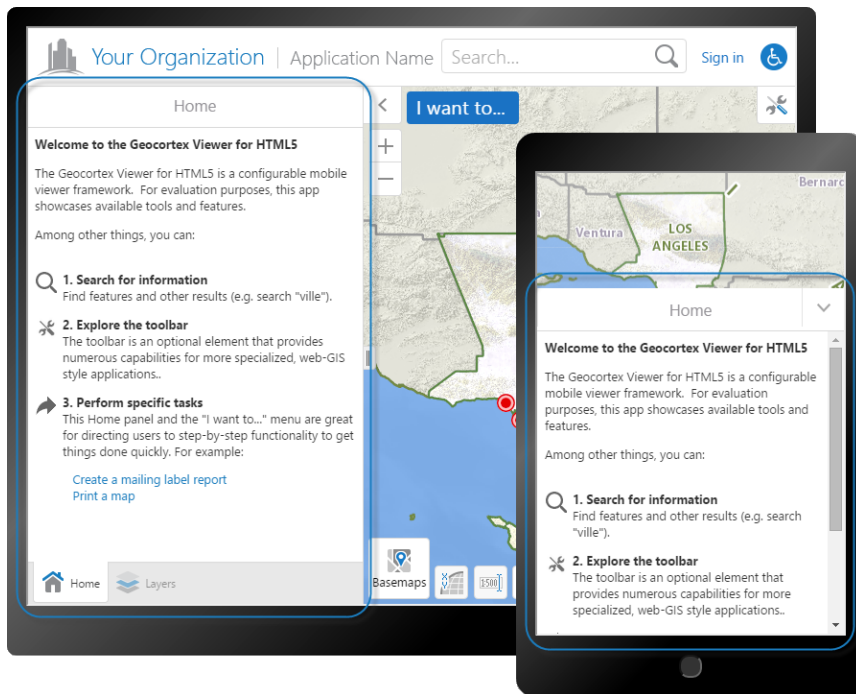
## 14.15 Configure the Home Panel

The Home Panel is a multipurpose panel with a variety of possible uses. You could use it to:

- Identify your organization.
- Publish a disclaimer.
- Introduce the application.
- Provide instructions for working with the application.
- Provide hyperlinks that run viewer commands or workflows. (See "Hyperlinks in the Viewer" in the *Geocortex Essentials Administrator Guide* for more information.)

You can configure the viewer to show the Home Panel when the user launches the viewer. In this case, the Panel serves as a welcome page.

The contents of the Home Panel are defined using HTML markup. The contents can include text, images, and UI elements like buttons and links that launch web pages, invoke commands, and run workflows. Initially, the Home Panel contains placeholder text to help give you ideas for how you might use the Panel.



Home Panel in the Desktop interface (rear), and in the Handheld interface

The Home Panel is implemented by the Info Module. See [Info Module on page 196](#) for information.

The Home Panel was introduced in version 1.3 of the HTML5 Viewer.

► **To open an HTML5 Viewer's Home Panel page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Home Panel**.  
The Home Panel page opens.

► **Before you configure settings on the Home Panel page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can always configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Home Panel page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Home Panel page has the following settings:

- **Include Home Panel:** When this checkbox is selected, the Home Panel is available in the viewer. If you do not want the viewer to have a Home Panel, clear the checkbox. By default, the viewer has a Home Panel.
- **Open Home Panel by Default:** When this checkbox is selected:
  - In the Handheld interface, the Home Panel displays instead of the map when the viewer launches.
  - In the Desktop and Tablet interfaces, the Data Frame has a tab for the Home Panel when the viewer launches. The user can click ► to see the Home Panel if the Data Frame is closed. Alternatively, the user can click the Home button 🏠 in the toolbar to see the Home Panel.



You can configure the Data Frame to open when the viewer launches by selecting the Open By Default checkbox on the Look and Feel page in Essentials.

By default, the Open Home Panel by Default checkbox is not selected.

If you clear the Open Home Panel by Default checkbox but leave the Include Home Panel checkbox selected, the user can click the Home button 🏠 in the toolbar to open the Home Panel.

- **Title:** The title that appears at the top of the Home Panel.





If your viewer is going to be available in more than one language, enter the text key that the Home Panel title is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

The default title is @language-common-welcome.



- **Content:** The HTML markup that defines the contents of the Home Panel. The Content box provides a Rich Text Editor with a toolbar at the top. By default, the Rich Text Editor is open in the Content box.


If the Rich Text Editor does not have a particular tool that you want, or you prefer to work directly in the HTML markup, click the Show Source icon  at the end of the toolbar. This displays the HTML markup in the Content box, so you can edit the HTML markup directly.

To return to the Rich Text Editor, click the  icon again. You can switch between the Rich Text Editor and the HTML markup as many times as you want.



You can use a valid text key for the content as long as there are no spaces or HTML markup surrounding the text key. For more information on using text keys, see [About User Interface Text on page 64](#).

### ► To configure the Home Panel:

1. On the **Home Panel** page, select the **Include Home Panel** checkbox.
2. If you want the Data Frame to have a tab for the Home Panel when the user launches the viewer, select the **Show Home Panel** checkbox.
3. In the **Title** box, type the title that you want to appear at the top of the Home Panel. You can use a text key or the literal text.
4. In the **Content** box, create the contents of the Home Panel.  
Use the Rich Text Editor that displays by default, or click the Show Source icon  to work in the HTML markup.
5. Click **Apply Changes**.
6. If you want the user to see the Home Panel when the Desktop or Tablet interface launches, you must configure the Data Frame to open:



The Handheld interface does not have a Data Frame; if you want the Home Panel to be the first thing the user sees when the viewer launches, all you have to do is select Include Home Panel and Show Home Panel.

- a. Click **Look and Feel** in the side panel.  
The Look and Feel page opens.
  - b. In the **Data Frame** area, select the **Open By Default** checkbox.
  - c. Click **Apply Changes**.
7. Click **Save Site**.

### See Also...

[About User Interface Text on page 64](#)

## 14.16 Configure Instant Search in the HTML5 Viewer

Before you configure Instant Search settings in a viewer, you should configure Instant Search in the site. For information, see the "Global Search" section in the *Geocortex Essentials Administrator Guide*.

The Instant Search page allows you to configure Instant Search settings for a viewer. In general, you should not need to change these settings—the default values have been adjusted to work for the speed people usually type, the amount of information people can realistically work with, and so on. The only setting you might want to change is Maximum Results, which limits the number of search results that are shown.



You can navigate between search hints by pressing the **up arrow** (↑) or **down arrow** (↓) keys.

### ▶ To open an HTML5 Viewer's Instant Search page in Manager:

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Instant Search**.  
The Instant Search page opens.

### ▶ Before you configure settings on the Instant Search page:

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can always configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Instant Search page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Instant Search page has the following settings:

- **Search Hints:** When this checkbox is selected, the viewer suggests search terms to the user. The user can click a hint to search for the term given in that hint. By default, the Search Hints checkbox is selected.  
When the Search Hints checkbox is cleared, the viewer does not display search hints.
- **Give Precedence to Nearby Results:** To rank search results by how close they are to the center of the current map and relevance to search terms, check this checkbox. To rank search results only by relevance to search terms, uncheck this checkbox. By default, this checkbox is checked.
- **Autocomplete Delay (ms):** The number of milliseconds to wait after the user stops typing before search hints are displayed.
- **Minimum Prefix length:** The number of characters the user must type before search hints are displayed.
- **Maximum Search Hints:** The maximum number of search hints to display.
- **Maximum Results:** The maximum number of search results to show.

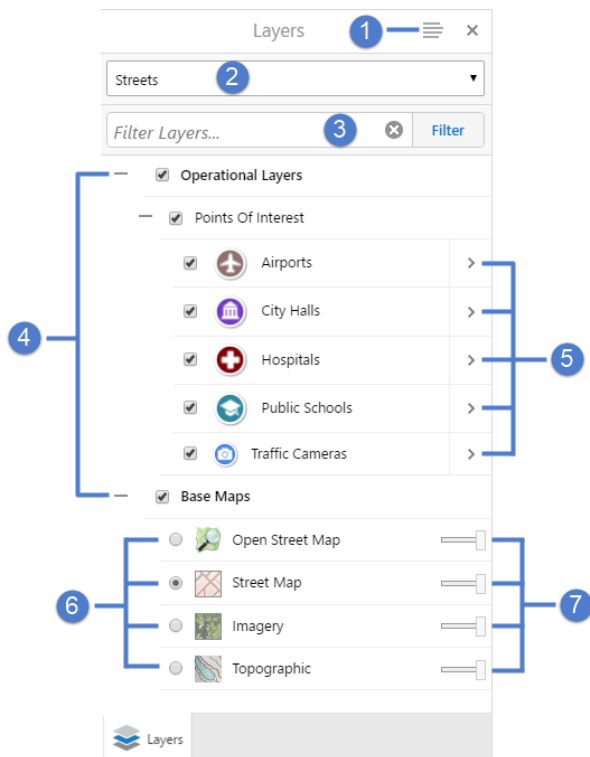
► **Before you leave the Instant Search page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

## 14.17 Configure the Layer List

The Layer List page in Manager has settings that control the appearance and behavior of the Map Layers panel that is displayed on the left side of the viewer. The Map Layers panel contains the configurable directory of folders, map services, layers and basemaps that are available in the viewer. It also contains a menu to select a layer theme.

You can use Manager to control which folders, map services and layers are initially turned on in the Map Layers panel when a user launches the map. For more information, see *The Layer List* in the *Geocortex Essentials Administrator Guide*.



**Layer List showing map services and layers**

1	Panel Actions menu	2	Layer Theme selection	3	Layer List Filter input field
4	Folders	5	Opens Layer Actions menus	6	Visibility settings
7	Transparency sliders				

► **To open an HTML5 viewer's Layer List page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Layer List**.  
The Layer List page opens.

► **Before you configure settings on the Layer List page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Geolocation page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Layer List page has the following settings:

- **Show Transparency Slider:** Select this checkbox if you want to display transparency sliders for each map service; otherwise, clear this checkbox. It is checked by default.
- **Include Legend Swatches:** Select this checkbox if you want to enable the ability to display legend swatches within the layer list; otherwise, clear this checkbox. It is checked by default.
- **Auto Expand Swatches:** Select this checkbox if you want to automatically display all legend swatches for each layer within the layer list; otherwise, clear this checkbox. It is unchecked by default.
- **Show Swatches For Visible Layers Only:** Select this checkbox if you want to only display legend swatches for layers that are visible; otherwise, clear this checkbox. It is unchecked by default.

► **Before you leave the Layer List page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

## 14.18 Configure Pushpins



Pushpins are not currently supported in the Handheld interface. Adding pushpin configuration to the Handheld configuration file has no effect.

The Pushpins page in Manager includes settings that control the appearance of pushpins that appear on the map. When pushpins are visible, they are placed in the center of a feature, subject to any configured offset.

Pushpins are visible for a point feature when:

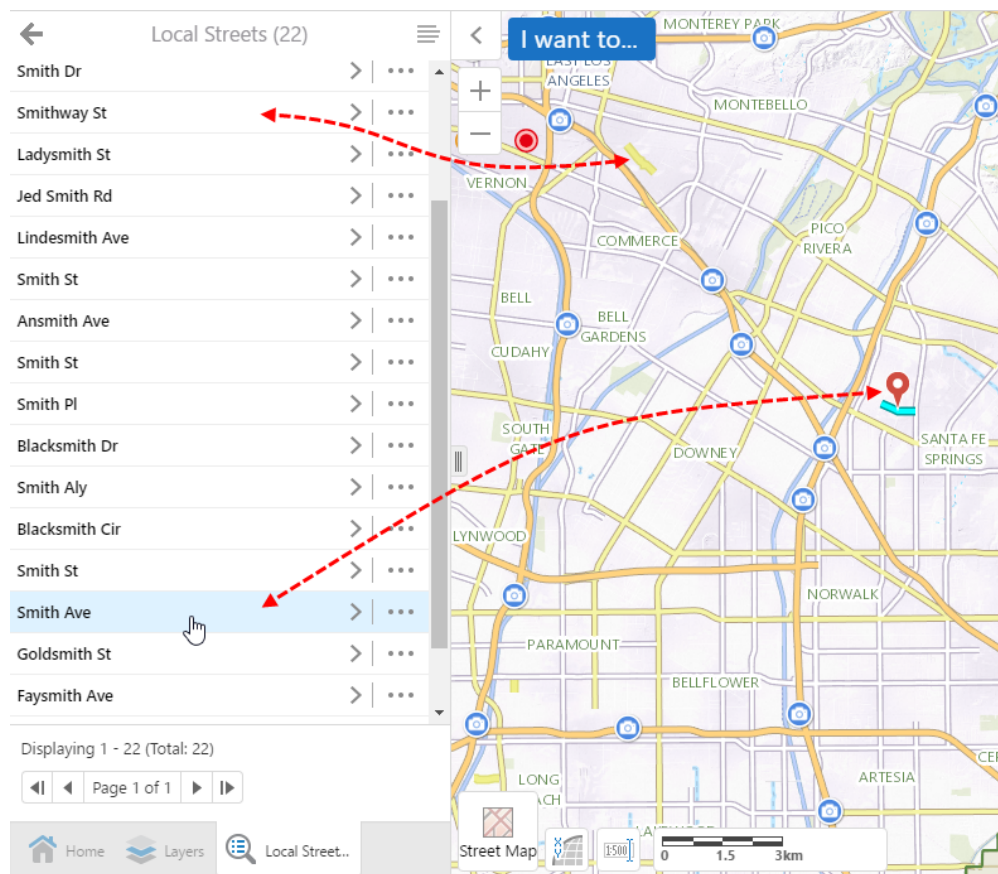
- The pointer is placed on the entry in the Results List.
- The feature details panel is open.
- A maptip for the feature is open.

Pushpins are visible for a line and/or polygon feature when:

- The feature is not visible in the current map scale and the pointer is placed on the entry in the Results List.
- The feature details panel is open.
- A maptip for the feature is open.

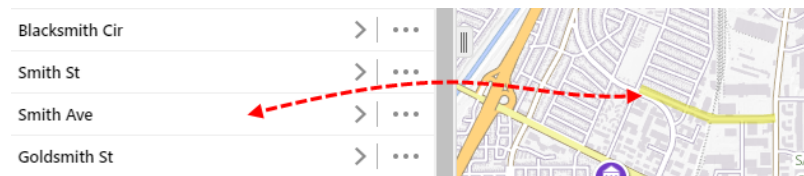
Otherwise, the feature is highlighted in yellow. If you place the pointer on a specific item in the list, the highlight color changes to blue on the map.

In the following example, a search for “smith” results in a list of street names, and their locations are highlighted in yellow on the map. Since the scale of the map does not show sufficient detail for the listed items, a pushpin identifies the location of the feature when the pointer is placed on that item in the list, and the highlight color changes to blue.

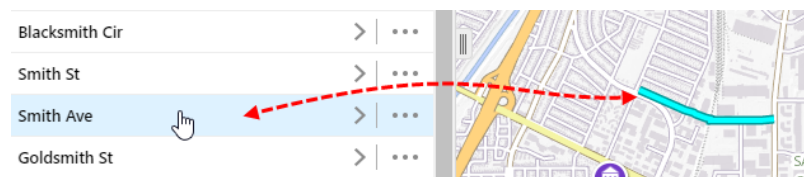


#### Highlighted line features and a pushpin on the map for the item in focus in the list


When the map zoom level is increased, the features remains highlighted in yellow. When focus is placed on the item in the list, the highlight color for that item changes to blue on the map, however a pushpin does not appear.




Feature on the map highlighted in yellow for the entry in the list



Feature on the map highlighted in blue when focus is on the entry in the list

 If you add markup to a map that has pushpins displayed, the markup appears behind the pushpins by design.

 Pushpins are supported with geocoders and workflows.

 **To open an HTML5 viewer's Pushpins page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Pushpins**.  
The Pushpins page opens.

 **Before you configure settings on the Pushpins page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Pushpins page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Pushpins page has the following settings:

- **Enabled:** If you want pushpins to appear for each feature of the search results, select this checkbox; otherwise, clear it. By default, this checkbox is selected.
- **Pushpins Remain Visible When Results List Inactive:** If you want pushpins to remain visible when a view other than the Results List or Results Table activates, select this checkbox; otherwise, clear it. By default, this checkbox is selected.

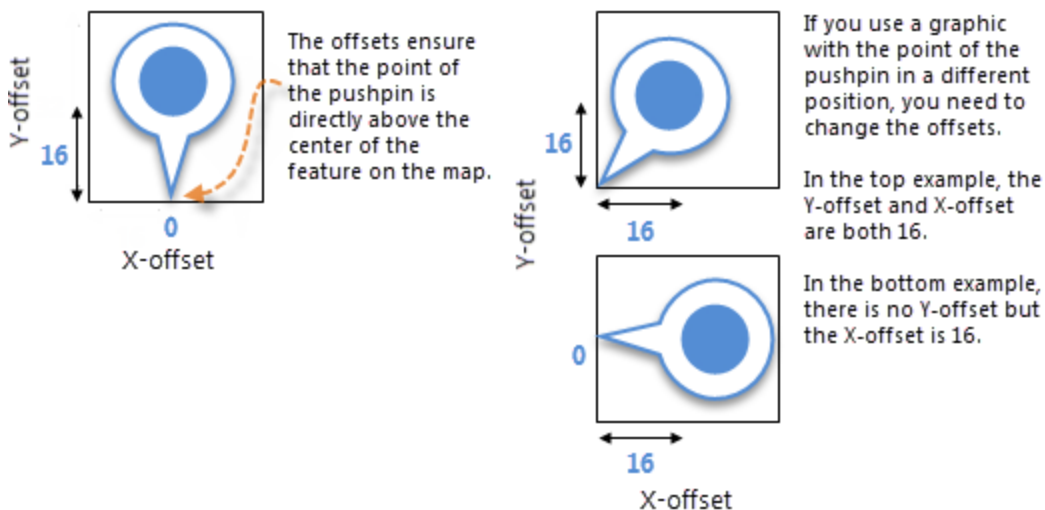


In versions prior to the HTML5 Viewer 2.6, the default behavior was to hide pushpins when a view other than the Results List or Results Table activates. If you want the previous default behavior, clear this checkbox.



This option has been deprecated effective with Essentials Manager 4.7 as a result of changes implemented in highlighting. For further information about highlighting, see [Highlight Module on page 191](#).

- **Pushpin Image:** The URL to the image that represents the pushpin. Type the URL in the text box. Alternatively, click **Browse** to use an image on the server; if necessary, upload an image to the server by clicking **Upload** and selecting an image file to upload. The default is a red pushpin image located at **Resources/Images/Pushpins/map-marker-red-32.png**. Other colors available out-of-the-box include blue, green, purple and yellow.
- **Pushpin Marker Width:** The width of the pushpin in pixels. This is typically set to the width of the pushpin image, otherwise the image will be scaled. The default is **32**.
- **Pushpin Marker Height:** The height of the pushpin in pixels. This is typically set to the height of the pushpin image, otherwise the image will be scaled. The default is **32**.
- **Offset X:** The number of pixels to horizontally offset the pushpin image. In other words, the distance along the X-axis between the center of the feature and the center of the pushpin image. Use a negative integer to place the pushpin to the left of the feature's center. The default is **0**.
- **Offset Y:** The number of pixels to vertically offset the pushpin image. In other words, the distance along the Y-axis between the center of the feature and the center of the pushpin image. Use a negative integer to place the pushpin below the feature's center. The default is **16**, because the pointer is at the bottom of the default image, which is 32 pixels high.



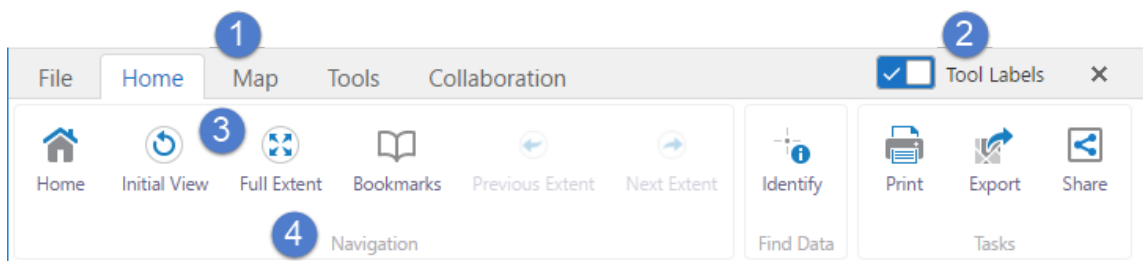
► **Before you leave the Pushpins page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

## 14.19 Configure the Toolbar

By default, a toolbar has a few basic tools in it to navigate the map, identify features, and print the map.

Also included by default on the Tabbed Toolbar is a toggle that allows the user to show or hide tool labels. The toggle is in the desktop and tablet shells only. It is not included in the Compact Toolbar.

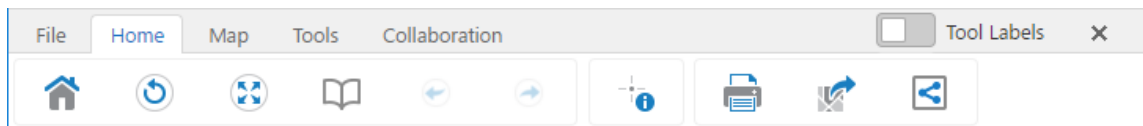


### Toolbar with the label toggle in the on position

1	Tab labels	2	Toolbar label toggle in the on position	3	Tool icons and labels	4	Tool group labels
---	------------	---	---	---	-----------------------	---	-------------------

Toggling the labels off:

- Hides the labels identifying the tools.
- Hides tool group labels.
- Reduces the font size of the text on toolbar tabs.
- Increases the display area for the map.



### Toolbar with the label toggle in the off position

Note that for keyboard users, when the toggle control button has focus, the Space Bar is the keyboard shortcut that toggles the control on or off.

► **To open a viewer's Toolbar page in Manager:**


1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Toolbar**.  
The Toolbar page opens.



## Before You Configure Settings on the Toolbar Page

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can always configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Toolbar page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

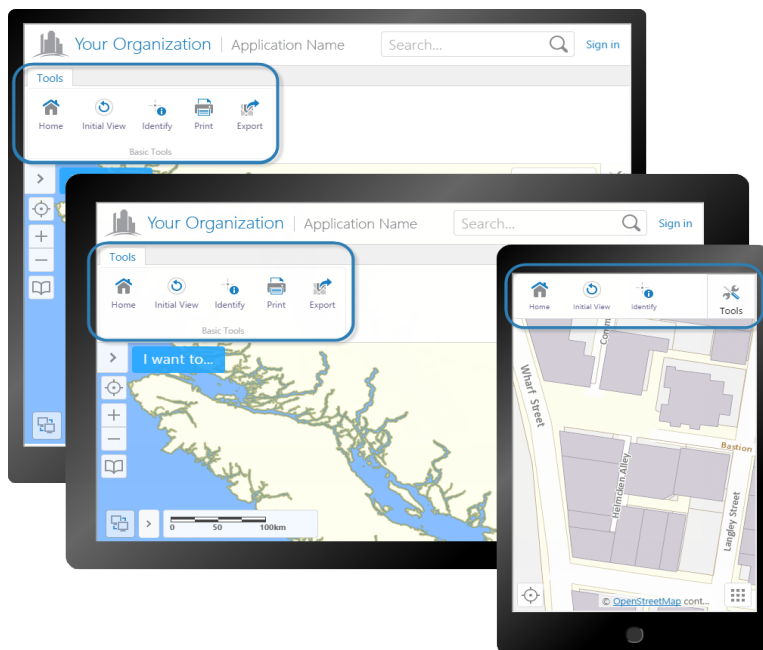
 By default, the Toolbar is configured individually because the Tabbed Toolbar is the default for the Desktop and Tablet interfaces, while the Compact Toolbar is the default for the Handheld interface.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Toolbar Types

There are two types of toolbars available:

- **Tabbed Toolbar:** Includes tabs and groups to organize tools in a particular order. This is the default toolbar for the Desktop and Tablet interfaces, and appears below the banner when activated. It may also be used in the Handheld interface.
- **Compact Toolbar:** Does not include tabs or groups and is designed to be simple and compact. This is the default toolbar for the Handheld interface, and appears at the top of the screen when activated. It may also be used in the Desktop and Tablet interfaces, where it appears below the banner.



The Tabbed Toolbar in the Desktop (back) and Tablet (middle) interfaces; and the Compact Toolbar in the

## Handheld interface


### ► To configure the type of toolbar to use:

1. Beside **Active Toolbar**, select either **Tabbed Toolbar** or **Compact Toolbar**.



To specify the number of tools to display at one time in the Compact Toolbar, type a number in **# of Tools to Display**.

## Toolbar Open/Closed State

By default, the toolbar is initially closed when the user launches the viewer. To open the toolbar, the user clicks the **Open toolbar** tool  in the upper right corner of the map. In the Handheld interface, if the map is not open, the user must open the map to see the toolbar icon. In the Desktop and Tablet interfaces, the user can also double-click the banner to open the toolbar.

### ► To open the toolbar when the viewer starts:


1. Check the **Open Toolbar by Default** checkbox.

## Tool Stickiness

Most tools in the HTML5 viewer are sticky by default—they remain active (selected) until deselected by the user. This allows the user to use a tool repeatedly without having to reselect it each time. You can turn off stickiness in the configuration. Each tool has a setting for stickiness, so you can make some tools sticky, and others not sticky.

### ► To disable or re-enable tool stickiness:

Stickiness applies to tools only. It does not apply to buttons.

1. Click the **Edit**  icon beside the tool.
2. To disable stickiness, clear the **Is Sticky** checkbox.
3. To re-enable stickiness, select the **Is Sticky** checkbox.

## Web GIS (Full) Preset Toolbar

The Web GIS toolbar contains a full set of tools grouped onto tabs that are designed for GIS implementations.

### ► To use a preset Tabbed Toolbar:

1. Click **Load Preset Toolbar**.
2. To use the default Tabbed Toolbar, click **Standard (Default)**. To use a comprehensive Tabbed Toolbar with nearly all tools, click **Web GIS (Full)**.



The **Web GIS (Full)** toolbar does not include Offline Management tools, as these tools require the Geocortex Mobile App Framework.


## Tool Types

You can add the following types of item to the toolbar:

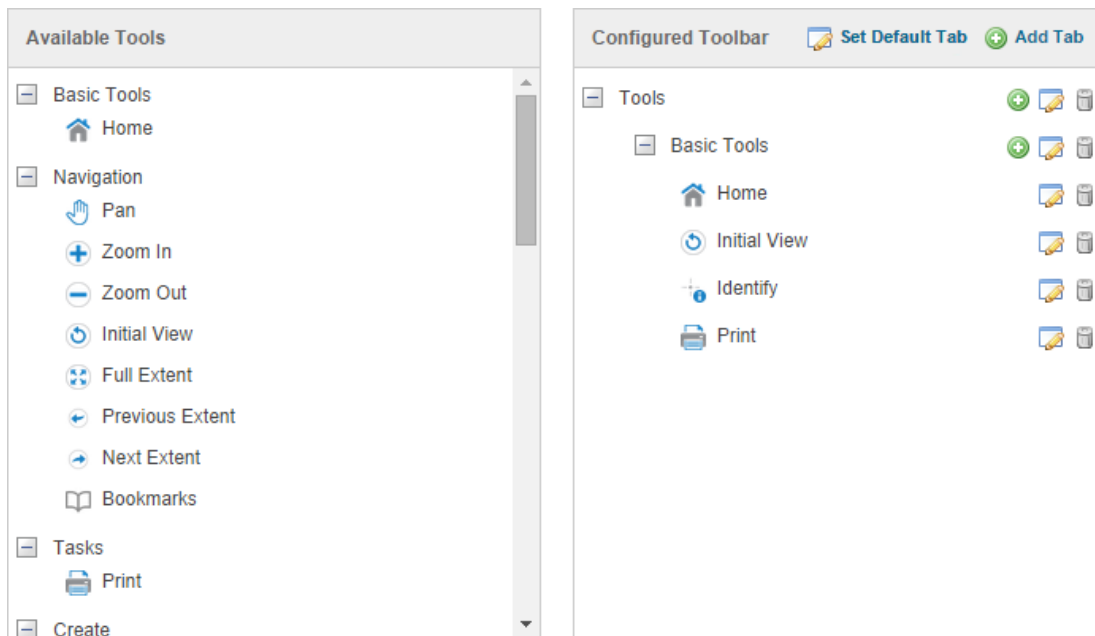
- **Buttons:** When clicked, a button immediately runs its command using the parameter that is configured for it, if it has one. For a list of viewer commands, see the [Geocortex SDK for HTML5 API Reference](#).
- **Toggle Buttons:** When clicked, a toggle button either turns on or off. Because of this, a toggle button can run two commands immediately when clicked: one when it is turned on, and another when it is turned off. When a toggle button is turned on, a selected checkbox will appear on the button. The commands may be different, or the commands may be identical but have different parameters. For a list of viewer commands, see the [Geocortex SDK for HTML5 API Reference](#). A toggle button may optionally be associated with a toggle state.
- **Tools:** When clicked, a tool waits for the user to draw a shape on the map, and then runs its command using the geometry created by the user as its parameter. For a list of viewer commands, see the [Geocortex SDK for HTML5 API Reference](#).
- **Multitools:** A multitool offers the user a menu of various related tools or buttons to use.
- **Regions:** A region is an area in the toolbar that contains custom content. For information on adding custom content to a toolbar region, contact Geocortex Support (<https://support.geocortex.com/>).

## Toolbar Contents

You can use the toolbar in its default configuration, or customize it by adding, modifying, and removing tools. For the Tabbed Toolbar, you can also reorganize the toolbar by adding, modifying, and deleting groups and tabs, and by moving tools to different groups or tabs.

To configure the toolbar, edit the viewer in Manager and click Toolbar in the side panel. The Toolbar page has two lists—a list of Available Tools on the left, and a list of tools in the Configured Toolbar on the right. To add a tool to the toolbar, drag the tool from the Available Tools list and drop it where you want it on the Configured Toolbar. To remove a tool from the toolbar, click the Remove icon .


Additionally, the toolbar is optional. You can turn it off using the Remove Toolbar function.




Drag and drop tools to customize the toolbar


► **To limit the number of tools displayed at one time in the Compact Toolbar:**

1. Type a number in **# of Tools to Display**. The minimum is **3** and the maximum is **9**. The default is **4**.

 This only applies to the Desktop and Tablet interfaces; it does not apply to the Handheld interface.

► **To add a tab to the Tabbed Toolbar:**


1. In the **Configured Toolbar** area, click  **Add Tab**.  
The Add Tab dialog box opens.
2. In the **Display Name** box, type the text to appear on the tab.

 If your viewer is going to be available in more than one language, instead of typing the literal text, type the text key that the tab display name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.


3. Click **OK**.  
The tab is added to the bottom of the Configured Toolbar area.
4. To move the tab to a different location in the toolbar, click the tab in the **Configured Toolbar** area and drag it to its new location.

▶ **To configure the Tabbed Toolbar's default tab:**

The first time in a user session that the user opens the toolbar, the toolbar opens to the default tab. If the user closes the toolbar and then reopens it, it opens to the tab that the user was last on.





1. In the **Configured Toolbar** area, click  **Set Default Tab**.
2. Select the tab that you want to open by default.
3. Click **OK**.

▶ **To add a group to a tab:**

1. In the **Configured Toolbar** area, click the  icon beside the tab to which you want to add the group.  
The Add Group dialog box opens.
2. In the **Display Name** box, type the name for the group. You can use a text key or the literal text.  
In the Desktop and Tablet interfaces, the name appears at the bottom of the group, under the group's buttons and tools. In the Handheld interface, the name appears at the top of the group.
3. Leave the **Multitool** checkbox cleared.
4. Click **OK**.  
The group is added as the last group in the tab.
5. To move the group to a different location in the toolbar, click the group in the **Configured Toolbar** area and drag it to its new location.

▶ **To add a multitool to a tab:**

A multitool acts as a menu of various related tools or buttons.

1. Use one of the following methods to add the multitool:
  - **Drag and Drop:**
    - a. Click the multitool  in the **Available Tools** panel, and then drag it to the **Configured Toolbar** panel. For example, the  **Draw** multitool.
    - b. Click the **Edit** button  beside the multitool in the **Configured Toolbar** panel.  
The Edit Multitool dialog box opens.
  - **Menu Option:**
    - a. In the **Configured Toolbar** area, click the  icon beside the tab to which you want to add the multitool.  
The Add Group dialog box opens.
2. In the **Display Name** box, type the name for the multitool. You can use a text key or the literal text.
3. Select the **Multitool** checkbox.




It is possible to convert the multitool into a group by unchecking the **Multitool** checkbox. While it is possible to include a multitool within a group, it is not possible to include a group within a group. A group can only be within a tab.

4. Click **OK**.

The multitool is added as the last item in the tab.

5. To move the multitool to a different location in the toolbar, click the multitool in the **Configured Toolbar** area and drag it to its new location.



To view and edit the items within a multitool, click the multitool icon .




A multitool cannot contain other multitools.

► **To add a button to a group:**


A button runs its command immediately when the user clicks the button. If the user needs to draw a geometry for the command to operate on, you must [add a tool](#) instead of a button.

1. Use one of the following methods to add the button:

• **Drag and Drop:**

- a. Click the button in the **Available Tools** panel, and then drag it to the **Configured Toolbar** panel.
- b. Click the **Edit** button  beside the button in the **Configured Toolbar** panel.  
The Edit Button dialog box opens.

• **Menu Option:**

- a. In the **Configured Toolbar** area, click the  icon beside the group that you want to add the button to.
- b. Click **Add Button**.  
The Add Button dialog box opens.

2. Configure the button's settings:

- **Name:** The name that you want to appear on the button. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-clear` or **Clear Markup**.
- **Tooltip:** The text for the tooltip that opens when the user positions the pointer over the button. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-clear-tooltip` or **Clear all drawings from the map**.

- **Hide on Disable:** When this checkbox is selected and the button's command cannot run under the current configuration or run-time conditions, the button does not show in the toolbar.

For example, if the Include Home Panel checkbox is cleared, a command to show the Home Panel cannot run because the viewer does not have a Home Panel. In this case, selecting the Hide on Disable checkbox for the Home Panel button hides the button.

If the checkbox is cleared and the button's command cannot run, the button shows in the toolbar, but it is grayed out.

- **Image URI:** The URI for the icon that you want to appear on the button. The image must be an appropriate size to fit on the button; for example, 24 x 24 pixels. Valid file formats are PNG, BMP, JPG, and JPEG.

To browse to the image file:

- Click **Browse**.  
The Select File dialog box opens.
- Expand the hierarchy in the left panel. When you can see the folder where you keep images for this viewer, open the folder.
- If the file has not yet been uploaded to the folder, click **Upload** and upload the file.
- Select the file.
- Click **OK**.

- **Command:** The command that the button runs when the user clicks the button. Click in the **Command** box to open a drop-down list of commands, and then select a command from the list.

For more information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).

- **Command Parameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.

For more information on a particular command's parameter, see [Geocortex SDK for HTML5 API Reference on page 422](#).

- Click **OK**.

The button is added as the last button in the group.


- To move the button to a different location in the toolbar, click the button in the **Configured Toolbar** area and drag it to its new location.

### To add a toggle button to a group:


A toggle button runs a command immediately when the user clicks it to turn it on, and a different command when the user clicks it to turn it off.

- Use one of the following methods to add the button:

- **Drag and Drop:**

- Click the button in the **Available Tools** panel, and then drag it to the **Configured Toolbar** panel.
- Click the **Edit** button  beside the button in the **Configured Toolbar** panel.  
The Edit Button dialog box opens.

- **Menu Option:**

- a. In the **Configured Toolbar** area, click the  icon beside the group that you want to add the button to.
- b. Click **Add Button**.  
The Add Button dialog box opens.

2. Configure the button's settings:

- **Toggle On Configuration:** Configures the toggle-on button, which turns the toggle button on.

- **Name:** The name that you want to appear on the toggle-on button. You can use a text key or the literal text.

For example, `@language-toolbar-home-sub` or **Home**.

- **Tooltip:** The text for the tooltip that opens when the user positions the pointer over the toggle-on button. You can use a text key or the literal text.

For example, `@language-toolbar-navigation-home-tooltip` or **Returns to introductory page**.

- **Hide on Disable:** When this checkbox is selected and the toggle-on button's command cannot run under the current configuration or run-time conditions, the toggle-on button does not show in the toolbar.

For example, if the Include Home Panel checkbox is cleared, a command to show the Home Panel cannot run because the viewer does not have a Home Panel. In this case, selecting the Hide on Disable checkbox for the toggle-on button hides the toggle-on button.

If the checkbox is cleared and the toggle-on button's command cannot run, the toggle-on button shows in the toolbar, but it is grayed out.

- **Image URI:** The URI for the icon that you want to appear on the toggle-on button. The image must be an appropriate size to fit on the toggle-on button; for example, 24 x 24 pixels. Valid file formats are PNG, BMP, JPG, and JPEG.

To browse to the image file:

- a. Click **Browse**.  
The Select File dialog box opens.
- b. Expand the hierarchy in the left panel. When you can see the folder where you keep images for this viewer, open the folder.
- c. If the file has not yet been uploaded to the folder, click **Upload** and upload the file.
- d. Select the file.
- e. Click **OK**.

- **Command:** The command that the toggle-on button runs when the user clicks the toggle button to turn it on. Click in the **Command** box to open a drop-down list of commands, and then select a command from the list.

For more information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).

- **Command Parameter:** The value for the command to use as its parameter, if it has a parameter.



Parameters may either be simple strings or complex objects containing any number of parameters. For more information on a particular command's parameter, see [Geocortex SDK for HTML5 API Reference on page 422](#).

- **Toggle Off Configuration:** Configures the toggle-off button, which turns the toggle button off.
  - **Name:** The name that you want to appear on the toggle-off button. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-clear` or **Clear Markup**.
  - **Tooltip:** The text for the tooltip that opens when the user positions the pointer over the toggle-off button. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-clear-tooltip` or **Clear all drawings from the map**.
  - **Hide on Disable:** When this checkbox is selected and the toggle-off button's command cannot run under the current configuration or run-time conditions, the toggle-off button does not show in the toolbar.  
For example, if the user hasn't drawn any markup, a command to clear the markup cannot run because no markup exists. In this case, selecting the Hide on Disable checkbox for the toggle-off button hides the toggle-off button.  
If the checkbox is cleared and the toggle-off button's command cannot run, the toggle-off button shows in the toolbar, but it is grayed out.
- **Image URI:** The URI for the icon that you want to appear on the toggle-off button. The image must be an appropriate size to fit on the toggle-off button; for example, 24 x 24 pixels. Valid file formats are PNG, BMP, JPG, and JPEG.  
To browse to the image file:
  - a. Click **Browse**.  
The Select File dialog box opens.
  - b. Expand the hierarchy in the left panel. When you can see the folder where you keep images for this viewer, open the folder.
  - c. If the file has not yet been uploaded to the folder, click **Upload** and upload the file.
  - d. Select the file.
  - e. Click **OK**.
- **Command:** The command that the toggle-off button runs when the user clicks the toggle button to turn it off. Click in the **Command** box to open a drop-down list of commands, and then select a command from the list.  
For more information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **Command Parameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters. For more information on a particular command's parameter, see [Geocortex SDK for HTML5 API Reference on page 422](#).

- **Associated State (Optional):**

- **Toggle State:** A optional toggle state to associate with the toggle button.



For a complete list of states, see the [State Reference on page 452](#).

3. Click **OK**.

The toggle button is added as the last button in the group.


4. To move the toggle button to a different location in the toolbar, click the toggle button in the **Configured Toolbar** area and drag it to its new location.

▶ **To add a tool to a group:**


A tool prompts the user to draw a geometry for the tool's command to operate on. If you want the command to run immediately when the user clicks it, you must [add a button](#) instead of a tool.

1. Use one of the following methods to add the tool:

- **Drag and Drop:**

- a. Click the tool in the **Available Tools** panel, and then drag it to the **Configured Toolbar** panel.
- b. Click the **Edit** button  beside the tool in the **Configured Toolbar** panel.  
The Edit Button dialog box opens.

- **Menu Option:**

- a. In the **Configured Toolbar** area, click the  icon beside the group that you want to add the tool to.
- b. Click **Add Tool**.  
The Add Tool dialog box opens.

2. Configure the tool's settings:

- **Name:** The name that you want to appear on the tool. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-rectangle` or **Draw Rectangle**.
- **Tooltip:** The text for the tooltip that opens when the user positions the pointer over the tool. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-rectangle-tooltip` or **Draw a rectangle on the map**.
- **Hide on Disable:** When this checkbox is selected and the tool's command cannot run, the tool does not show in the toolbar.  
If the checkbox is cleared and the tool's command cannot run, the tool shows in the toolbar, but it is grayed out.
- **Image URI:** The URI for the icon that you want to appear on the tool. The icon should be sized to fit on the tool. The image must be an appropriate size to fit on the button. Valid file formats are PNG, BMP, JPG, and JPEG.

To browse to the image file:

- a. Click **Browse**.  
The Select File dialog box opens.
- b. Expand the hierarchy in the left panel. When you can see the folder where you keep images for this viewer, open the folder.
- c. If the file has not yet been uploaded to the folder, click **Upload** and upload the file.
- d. Select the file.
- e. Click **OK**.

- **Command:** The command that the tool runs after the user has drawn the geometry for the command to operate on. Click in the **Command** box to open a drop-down list of commands, and then select a command from the list.

For more information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).

- **Draw Mode:** The type of geometry that the user will draw.
- **Status Text:** Optional text to display on the map to provide guidance to the user. You can use a text key or the literal text.

For example, `@language-toolbar-markup-rectangle-desc` or **Click and drag on the map to draw a rectangle on the map**.

- **Is Sticky:** When this checkbox is selected, tools remain active (selected) until deselected by the user—the tools are "sticky". This allows the user to use a tool repeatedly without having to reselect it each time.


If you do not want a tool to remain selected after it is used, clear the Is Sticky checkbox.

3. Click **OK**.  
The tool is added as the last tool in the group.
4. To move the tool to a different location in the toolbar, click the tool in the **Configured Toolbar** area and drag it to its new location.

#### To add a region to the toolbar:




To add content to a region that you created in Manager, you must edit the viewer's configuration file.

1. In the **Configured Toolbar** area, click the **Add Toolbar Item** icon  beside the group that you want to add the region to.  
A drop-down list opens.
2. Click **Add Region**.  
The Add Region dialog box opens.
3. **Region Name:** Type a name for the region.  
The name is visible in Manager only—it is not visible in the viewer.
4. Click **OK**.

The region appears at the bottom of the group.

5. To move the region to a different location in the toolbar, click and drag it.

▶ **To edit a tab, group, or toolbar item:**

1. Click the **Edit** icon  next to the item you want to modify.  
The Edit dialog box opens.
2. Edit the item as desired.
3. Click **OK**.

▶ **To change the order of items in the toolbar:**

You can move tabs, groups, buttons, tools and regions within the toolbar.

1. Click and hold the item you want to move.
2. Drag the item to its new location in the toolbar.

▶ **To revert to the default toolbar:**



Reverting to the default toolbar permanently deletes your custom toolbar.

Follow the instructions that apply to your type of toolbar:

• **Tabbed Toolbar:**

1. Click **Load Preset Toolbar**.
2. Click **Standard (Default)**.

• **Compact Toolbar:**

1. Click the **Revert to Default Toolbar** button.

The configuration for the default toolbar appears in the Configured Toolbar area.

▶ **To remove the toolbar:**

If you do not want the viewer to have a toolbar, you can remove the toolbar.



Removing the toolbar permanently deletes any customizations you have made to the toolbar.

1. Click the **Remove Toolbar** button.  
You are prompted to confirm.
2. Click **OK**.

► **To add back the toolbar if you have removed it:**

Follow the instructions that apply to your type of toolbar:

- **Tabbed Toolbar:**

1. Click **Load Preset Toolbar**.
2. Click **Standard (Default)**.

- **Compact Toolbar:**

1. Click the **Revert to Default Toolbar** button.

The configuration for the default toolbar appears in the Configured Toolbar area.

► **Before you leave the Toolbar page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

**See Also...**

[About User Interface Text on page 64](#)

## 14.20 Configure Tool Behavior

The Tool Behavior page in Manager contains settings that define how the Identify, Buffer and Measurement Tools operate in each shells of the HTML5 Viewer. In addition to controlling which layers are affected by the Identify tool, you can set the projection ID and default units to use for buffering and measuring distances and areas. It is also possible to set the type of calculation to use for measurement, and select whether or not to add the markup to the map automatically, or to enable prediction.

► **To open the Tool Behavior page for the HTML5 viewer in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Tool Behavior**.  
The Tool Behavior page opens.

► **Before you configure settings on the Tool Behavior page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can always configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Tool Behavior page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Tool Behavior page has the following settings:

### Identify Tools:

- **Pixel Tolerance:** The maximum number of pixels away from a feature the user can click with a point-based Identify tool in order to identify the feature. The default is **5**.



This does not apply to polygon-based identify operations, which can be manually modified via the `polygonPixelTolerance` property in the configuration file.

- **Visible Layers Only:** To only include layers marked as visible in identify operations, select this checkbox; otherwise, clear this checkbox. By default, this checkbox is selected.
- **Layers in Visible Scale Range Only:** To only include layers within the visible scale range in identify operations, select this checkbox; otherwise, clear this checkbox. By default, this checkbox is selected.

**Buffer:** The buffering tool performs geodesic buffering.



Buffering requires an ArcGIS Server 10.1 or newer geometry service.

- **Projection WKID:** Add the well-known ID (WKID) of the projection to use, or leave the field blank to disable projection. By default, projection is disabled.
- **Buffer Units:** Select the distance units to be available for buffering. Possible values include: **Feet (ft)**, **Yards (yd)**, **Meters (m)**, **Kilometers (km)**, **Miles (mi)**, and **Nautical Miles (NM)**. By default, all units are selected except for **Yards (yd)**.
- **Default Unit:** Select the default unit of distance to use for buffering. The default is **Kilometers (km)**.
- **Add as Drawing by Default:** If this checkbox is selected, any buffering is added to the map as markup by default.

### Measurement:

- **Projection WKID:** Add the WKID of projection for the Measurement module to use, or leave the field blank to disable projection. By default, projection is disabled. This setting only applies if the Calculation Type is set to Planar.
- **Default Length Unit:** Select the default unit of measurement to use for length. Possible values include **Feet (ft)**, **Yards (yd)**, **Meters (m)**, **Kilometers (km)**, **Miles (mi)**, and **Nautical Miles (NM)**.
- **Default Area Unit:** Select the default unit of measurement to use for area. Possible values include **Feet<sup>2</sup> (ft<sup>2</sup>)**, **Yard<sup>2</sup> (yd<sup>2</sup>)**, **Meter<sup>2</sup> (m<sup>2</sup>)**, **Kilometer<sup>2</sup> (km<sup>2</sup>)**, **Mile<sup>2</sup> (mi<sup>2</sup>)**, **Nautical Mile<sup>2</sup> (NM<sup>2</sup>)**, **Acres (ac)**, and **Hectares (ha)**.
- **Calculation Type:** Select the calculation type to use when calculating measurements.



Geodesic measurement requires an ArcGIS Server 10.1 or newer geometry service.

- **Planar:** Use Planar if you want to specify a particular WKID and you want to ensure that this projection is used no matter where in the world the measurements are taken. Planar strictly enforces a particular WKID. If you enter the incorrect WKID for a particular area, you can get inaccurate measurements. Planar measurements use 2D Cartesian mathematics to calculate area and length.



If you select the Planar calculation type and do not specify a Projection WKID, no projection takes place and measurements are computed in the default spatial reference of the site.



Essentials only allows a Projection WKID to be selected for the Planar calculation type. A Projection WKID is not required for the Geodesic or Preserve Shape calculation types.

- **Geodesic:** Use Geodesic if you intend to measure very long distances that require accounting for the curvature of the earth. The Geodesic calculation type computes distances, areas and perimeters using only the vertices of the polygon to define the lines connecting the vertices as geodesic segments, independently of the actual shape of the polygon. A geodesic segment is the shortest path between two points on an ellipsoid. The results, when you use a geodesic calculation type, are independent of the input spatial reference.
- **Preserve Shape:** Use Preserve Shape if you are unsure about which calculation type to use, as it will provide the most accurate results for most users. This calculation type computes distances, areas and perimeters on the surface of the Earth ellipsoid, while preserving the shape of the geometry in its coordinate system. This means that the true area and length will be calculated for the geometry that is displayed on the map. This is the default calculation type.
- **Add as Drawing:** Defines whether or not markup is added to the map by default. The default is `true`.
- **Prediction Enabled:** Defines whether or not the current segment length, total segment length, perimeter, and area are predicted when the cursor hovers over the map for one second. When set to `false`, it disables predictive measurements. The default value is `true`. Prediction makes it possible to see approximately what measurements you would get if you clicked to plot a vertex at a given point.

#### ► Before you leave the Tool Behavior page:

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

#### See Also...

[Configure the Toolbar on page 108](#)

## 14.21 Configure Optimizer Integration

Geocortex Optimizer captures and organizes information about your ArcGIS Server sites and infrastructure. If you have purchased a Geocortex Optimizer license, you can integrate your Geocortex viewers with Optimizer. When a viewer is integrated with Optimizer, the viewer sends usage data to Optimizer's Client API Relay collector. The data is then written

to the Optimizer database so you can run reports against it. The data includes information about the viewer's users, the layers and regions that they look at, and the tools that they use.

The Optimizer Integration page in Manager enables you to configure the settings required to send data to Optimizer. If you have not purchased Geocortex Optimizer or you are not using the Client API Relay collector, skip this page. To find out more about Geocortex Optimizer, contact your Geocortex Account Manager.

► **To open an HTML5 viewer's Optimizer Integration page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Optimizer Integration**.  
The Optimizer Integration page opens.

► **Before you configure settings on the Optimizer Integration page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Optimizer Integration page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Optimizer Integration page has the following settings:

- **Enabled:** When this checkbox is selected, the Viewer sends data to Optimizer. If you do not use Optimizer, clear this checkbox. By default, this checkbox is cleared.
- **Username:** A phrase indicating that the real username is not available because the user is not signed in. The default is **DefaultUser**.
- **Data Relay URI:** The URI to the Optimizer endpoint responsible for collecting data. If the endpoint is not specified, the Viewer attempts to log back to the same host that the Viewer application is launched from.

► **Before you leave the Optimizer Integration page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

## 14.22 Configure Analytics Integration

Geocortex Analytics collects data about and reports on GIS infrastructure. If you do not use Geocortex Analytics, skip this page.



If you are running Geocortex Analytics, you must configure your Viewer applications to send data to the Client Relay Collector in Analytics so it can report on viewer activity. The `InsightIntegration` module in Geocortex Viewer for HTML5 collects and sends usage and other data about the viewer to the Client Relay Collector at configurable intervals. The Analytics Integration page in Manager enables you to configure the settings required to send this data to Analytics.

▶ **To open an HTML5 viewer's Analytics Integration page in Manager:**

1. In Manager, edit the viewer that you want to configure.
2. In the side panel, click **Analytics Integration**.  
The Analytics Integration page opens.

▶ **Before you configure settings on the Analytics Integration page:**

Consider whether the settings will be the same for Desktop, Tablet, and Handheld interfaces:

- If you are not sure, keep the settings the same by configuring the interfaces together—you can configure them separately later without losing any changes you have made.
- If you are sure you want the settings on the Analytics Integration page to be different, click the **Configure Individually** hyperlink. You can switch back to configuring the interfaces together at any time, but you may lose some of the changes you have made.

For more information, see [About Configuring Multiple Interfaces on page 62](#).

## Settings

The Analytics Integration page has the following settings:

- **Enabled:** When this checkbox is selected, the Viewer sends data to Analytics. If you do not use Analytics, clear this checkbox. By default, this checkbox is cleared.
- **Data Relay URI:** The URI to the Analytics Client Relay Collector responsible for collecting data. Be sure to use the correct URI to the Client Relay, for example,  
`http://analyticshub.domainname.com/Geocortex/Analytics/ClientRelay`

▶ **Before you leave the Analytics Integration page:**

1. Click **Apply Changes**.
2. Use the Preview hyperlinks to see your configuration.  
See [About the Live Preview on page 64](#) for instructions.
3. To save your changes, click **Save Site**.

## 14.23 Configure Collaboration

Collaboration helps an organization share text messages, images, and markup in real time, directly in their mapping applications.

You can enable the **Collaboration** tab in the Toolbar in the site's viewer by configuring the collaboration options in Manager.



Collaboration is not supported in Internet Explorer version 9. Use Internet Explorer version 10 or later.

### ▶ To enable Collaboration:

1. In Manager, edit the viewer that you want to configure.
2. Click **Toolbar** in the side panel:
3. Locate **Collaboration** in **Available Tools** list.
4. Drag it from **Available Tools** and drop it where you want it on the **Configured Toolbar** list.
5. Click **Apply Changes** and then **Save Site** to save the changes to the toolbar before proceeding.
6. In the side panel, click **Collaboration** and change the **Mode** option from **Disabled** to **Live Collaboration**.
7. Click **Apply Changes** and then **Save Site**.

### ▶ Complete the setup for Collaboration in the viewer:

1. Sign in to the viewer if you have not already done so.
2. Open the Toolbar:
  - a. If you added Collaboration to the toolbar as a tab, locate and click the **Collaboration** tab and then click the **Collaboration** icon to open the panel.
  - b. If you added Collaboration to an existing toolbar tab, locate and click the **Collaboration** icon on that tab to open the panel.

The **Collaboration** panel is initially empty.

3. Click the **Join a Room** drop down and then click **Create a Room**.

Any user can create a room.
4. Type a name for the room.
5. Select a color identifier.

The color identifier helps to distinguish which messages in the Collaboration panel apply to each room.
6. Under **Permissions**, select **Individuals** or **Groups**, begin typing an individual or group name in the **Search** field, select a user or group in the filtered list, then click **Add**.

The user or group is included in the list under the permission settings with the **View** permission set by default.
7. Select user permissions from among **None**, **View**, **Edit**, or **Admin**.

User permissions can be assigned to any users that sign in to the room (**Signed-in Users**), and also separately for individuals or groups that have been included in the room.

The user creating the room is automatically included in the list of users as the administrator for that room. An administrator cannot edit their permissions (set to **Admin** by default). Only another user with administrative privileges can edit permissions for other administrators.

8. Click **Save** to save the new room.  
The room is immediately available for messages, drawings, or images.

▶ **To send a message to room participants:**

1. Type your message in the message entry area.
2. Click **Send**.

See [Working with Multiple Rooms on page 128](#) for additional information about sending messages.

▶ **To add a drawing:**

1. Click **Drawing** above the message entry area.
2. Select one of the available drawing tools.
3. To change the styling for the selected drawing tool, click **Style** and change any of the color, pattern, and other styling attributes.
4. Draw on the map.
5. Add any applicable comments in the message entry area.
6. To clear the drawing and/or the related message before sending it, click the **X** beside the **Drawing** heading. Click **OK** to confirm the action.
7. Click **Send** to post your drawing reference and text to the room for others to view.

To identify the drawing on the map associated with a message, click the message to pulse the drawing on the map. When you click a message for a drawing that is not in view or only partially visible on the map, the drawing on the map is centered. Depending on the zoom level for the map, the drawing is centered and zoomed in on, or it is centered without zooming.

▶ **To place an image:**

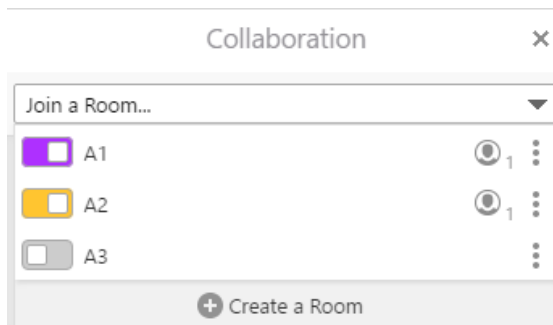
1. Click **Image** above the message entry area.
2. Select an image to upload.  
The locate option is automatically enabled.
3. Click the location on the map to apply the image icon.
4. To move the image icon, click **Locate**, and then drag the icon to another location.
5. Add any applicable comments in the message entry area.
6. To clear the message and/or the image before sending it, click the **X** beside the **Image** heading above the message entry area. Click **OK** to confirm the action.
7. Click **Send** to post your image attachment and text for others to view.

To identify the image icon on the map associated with a message, click the message to pulse the image icon on the map. When you click a message for an image icon that is not in view on the map, the image icon is centered in the map view.

Depending on the type of message, the menu in the upper right includes one or more of the options to **Erase**, **Copy to another room**, or **Edit** the contents of the message.

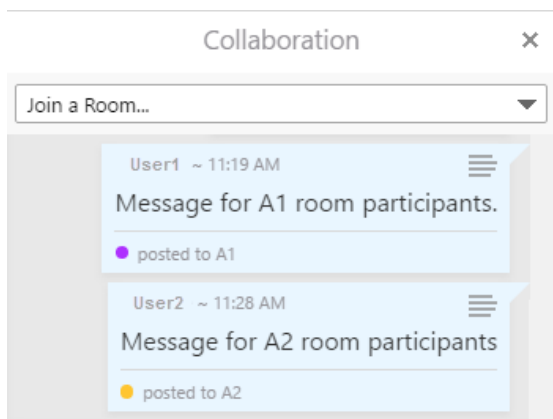
### 14.23.1 Working with Multiple Rooms

Each user can view the rooms that they are members of when they click **Join a Room**. The slider beside the room names allows a user to display or hide messages for that room. Clicking anywhere on the line toggles the option on or off.



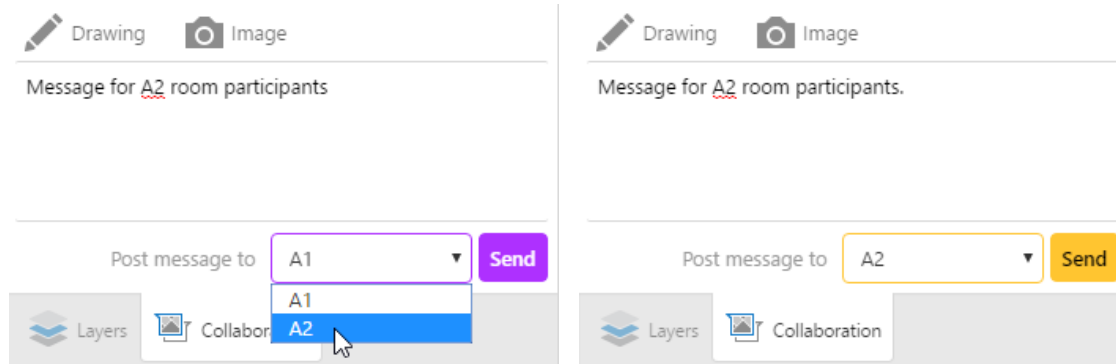
**Messages for rooms A1 and A2 are displayed. Messages for room A3 are not displayed**

When a user joins a room, the messages for that room are displayed. Each message contains a color identifier in the lower left to indicate the room that the message is associated with.



**Posted messages with the room color identifier in the lower left**

If the user joined multiple rooms, and has typed a message, they select the target room for the message from the **Post message to** list at the bottom of the message entry field. When the user selects a room, the color of the **Send** button changes to match the color identifier for the room. Note that only those rooms that the user has edit permissions for are in the **Post message to** list.



Selecting the room for the message

## 15 Configuration Settings by Module

This section lists the configuration properties for modules, views, and view models.

All modules have the properties listed in [Common Settings for Modules on page 129](#). All views have the properties listed in [Common Settings for Views on page 130](#). All view models have the properties listed in [Common Settings for View Models on page 130](#).

In addition, a module can have additional module, view, and view model properties that are specific to that module. These additional properties are listed in the section for that module. For example, [Banner Module on page 134](#) lists the properties that are specific to the Banner Module. The Banner Module also has module, view, and view model properties that are common to all modules, views, and view models respectively.

See also...

[Application-Wide Settings on page 58](#)

### 15.1 Common Settings for Modules

Each module in the `modules` section of a configuration file has the following properties:

- **moduleName:** The name of the module .
- **moduleType:** The type of module.
- **libraryId:** (optional) The ID of the library to download. If `libraryId` is not specified, the module downloads the default library.
- **configuration:** Module-specific configuration that is passed to the module when it is initialized. If there is no module-specific configuration for a particular module, its `configuration` property is empty.  
For information on the configuration for a specific module, see the section for that module in [Configuration Settings by Module on page 129](#).
- **views:** (optional) An array of views of the module.
- **viewModels:** (optional) An array of view models of the module.

## 15.2 Common Settings for Views

Each view specified in a configuration file has the following properties:

- **id:** The view's unique ID. The ID is used to reference the configured view, for example when activating it using a view command such as `ActivateView`.
- **viewModelId:** (optional) The configured view model that the view is to be bound to. Views with a view model ID are attached to the referenced view model as soon as the view model is instantiated and initialized. Configured views without a view model ID have an anonymous (blank) view model attached.
- **visible:** (optional) Specifies whether or not the view should be activated when it is first hosted in a region. The default is `false`.  
A view with the visible flag set to `true` in configuration is activated by the view manager when it is created. This causes the host region to activate the view. The behavior of view activation depends on the region adapter used, but generally an activated view becomes visible and interactive.
- **markup:** The path to use to render the view. The view markup is created and added to the DOM when the view is hosted or activated by its host region. This path represents the relative file system path where the HTML resource was found when compiled, based on the compilation root specified when running the resource compiler tool.
- **region:** The region that the view is hosted in. If the referenced region does not exist at the time of view creation, the view is considered to be pending—it is added to the region when the region becomes available. See [Regions on page 444](#) for a list of regions.
- **isManaged:** (optional) Specifies whether a view is to be managed or not. The meaning of "managed" is open-ended. A component that displays a list of views or allows a user to activate and deactivate views can choose to observe this setting or not. This allows views to opt out of being displayed in components such as window managers, docks, or toolbars that show active views.
- **configuration:** View-specific configuration that is passed to the view when it is initialized. If there is no view-specific configuration for a particular view, its `configuration` setting is empty.  
For information on the configuration for a specific view, see the section for the module the view belongs to, in [Configuration Settings by Module on page 129](#).

## 15.3 Common Settings for View Models

Each view model specified in a configuration file has the following properties:

- **id:** The view model's unique ID, used to reference the view model. Configured views have a `viewModelId` property that references the view model ID.
- **type:** The type of view model. This is used by the application to instantiate the view model.
- **configuration:** View model-specific configuration that is passed to the view model when it is initialized. If there is no view model-specific configuration for a particular view model, its `configuration` setting is empty. For information on the configuration for a specific view model, see the section for the module the view model belongs to, in [Configuration Settings by Module on page 129](#).


## 15.4 Accessibility Module

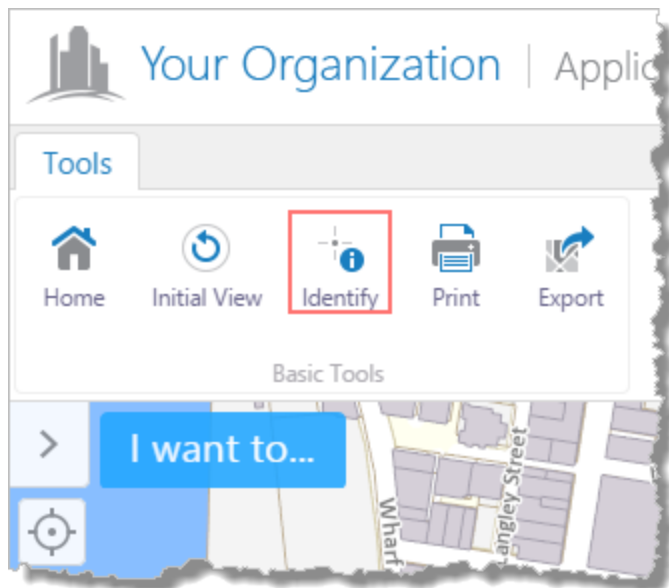
The Accessibility Module implements accessibility features that make the HTML5 Viewer easier to use for people with disabilities. There are two aspects to accessibility support in the HTML5 Viewer that users can use:

- **Screen Readers:** Run a screen reader to vocalize and interpret page content.
- **Keyboard Shortcuts:** Interact with the viewer using only the keyboard.

Screen readers and keyboard shortcuts can be used together.

The Accessibility Module also implements the configurable [Accessibility Panel](#), which informs users about the accessibility features in the HTML5 Viewer.

 In order for a user to use a screen reader with an HTML5 Viewer, the user must have a screen reader installed and running when using the viewer. No additional steps are required. The HTML5 Viewer is tested using the Freedom Scientific [JAWS screen reader](#) but others may also work.



The user presses TAB repeatedly to navigate to the Identify tool

## Configuration Properties

### Module

- **keyboardFocusIndicatorEnabled:** To highlight the current UI element with a border, set to `true`; otherwise, set to `false`. The default is `true`.
- **keyboardFocusIndicatorColor:** If the `keyboardFocusIndicatorEnabled` configuration property is set to `true`, use this setting to change the highlight color of the current UI element. The default is `#550055`.

- **expandedMapKeyboardAccessibility:** To allow the user to pan the map with the arrow keys even when the mouse pointer is not hovering over the map, set to `true`; otherwise, set to `false`. The default is `true`.



The user must navigate to the map before panning with the arrow keys.

- **automaticElementFocusing:** To automatically focus on the first interactive element of a newly-activated view, set to `true`; otherwise set to `false`. The default is `true`.
- **includeProviders:** To enable all of the accessibility providers that read out information to users with screen readers, set to `true`; otherwise set to `false`. The default is `true`.
- **providers:** An array of accessibility providers. By default, they are `MapTextProvider` and `ViewActivatorProvider`:
  - **MapTextProvider:** Provides the ability to read what is currently visible on the map to the user. This provider has the following properties:
    - **id:** The ID of the provider.
    - **type:** The provider type.
    - **decimalPrecision:** The number of decimal places to read out. The default is 4.
    - **readAttributionInformation:** To have the screen reader read aloud the map attribution information when the map is selected, set to `true`; otherwise, set to `false`. The default is `false`.
    - **isEnabled:** To enable the provider, set to `true`; otherwise, set to `false`. The default is `true`.

**ViewActivatorProvider:** Provides the ability to read out which views have been activated or deactivated. This provider has the following properties:



- **id:** The ID of the provider.
- **type:** The provider type.
- **isEnabled:** To enable the provider, set to `true`; otherwise, set to `false`. The default is `true`.
- **notifications:** An object whose properties' names are non-Data-Frame views, and values are text keys or strings representing the name of the view to read out when the view is activated or deactivated.
- **subviewNotifications:** An object whose properties' names are Data-Frame views, and values are text keys or strings representing the name of the view to read out when the view is activated or deactivated.

## Views

- **AccessibilityView:** None
- **AccessibilityIconView:** None



## View Models

- **AccessibilityViewModel:** None
  - **AccessibilityIconViewModel:**
    - **included:** To display the **Accessibility Button**  in the Desktop or Tablet interfaces, or the **View the accessibility panel** option in the [I Want To Menu](#) in the Handheld interface, both of which open the Accessibility Panel, set to `true`; otherwise, set to `false`. The default is `true`.
    - **content:** The encoded HTML content of the Accessibility Panel.
-  You can use a valid text key for the content as long as there are no spaces or HTML markup surrounding the text key. For more information on using text keys, see [About User Interface Text on page 64](#).
- **title:** The title of the Accessibility Panel. You can specify a string or a language key. The default is the language key, `@language-accessibility-map-title`, which reads **Accessible Geocortex**. For more information on using text keys, see [About User Interface Text on page 64](#).

### See Also...

[Accessibility on page 402](#)

[Configure Accessibility on page 72](#)

[IWantToMenu Module on page 215](#)

[About User Interface Text on page 64](#)

## 15.5 Alert Module

The Alert Module displays customized alerts.

## Configuration Properties

### Module

- **alertRegion:** The region in which to display the alert. The default region is `ModalWindowRegion`. For a list of regions, see [Regions on page 444](#).
- **overrideNativeAlert:** When `overrideNativeAlert` is set to `true`, the browser's native alert method is overridden. By default, `overrideNativeAlert` is `true`. Calling the `alert` method with `message`, `title`, and `callback` invokes the custom alert.

### Views

- **AlertView:** None

## View Models

None

## 15.6 Authentication Module

The Authentication Module creates and presents a form for users to enter their credentials and be authenticated. You can configure the region in which the form is rendered.

## Configuration Properties

### Module

- **region:** The region in which to display the authentication form. The default region is `ModalWindowRegion`. For a list of regions, see [Regions on page 444](#).

### Views

- **AuthenticationView:** None

### View Models

- **AuthenticationViewModel:** None

## 15.7 Banner Module



This module can be configured using Manager. For instructions, see [Change the Look and Feel on page 74](#).

The Banner Module displays the area at the top of the viewer that contains the viewer's title. The Handheld interface does not have a banner.

## Configuration Properties

### Module

None

### Views

- **BannerView:** None
- **UserInfoView:** None
- **SignInView:** None

- **SearchView:** None
- **SearchHintsView:** None

## View Models

- **BannerViewModel:**
  - **applicationTitle:** The text of the title to display in the banner. If you want control over the font and size, leave `applicationTitle` empty, embed the title in an image with the logo, and specify the image using one of the image properties. The title is optional.
  - **applicationSubtitle:** The subtitle to display in the banner. The subtitle is the smaller text that appears under the title. The subtitle is optional.  
If you want control over the font and size, leave `applicationSubtitle` empty, embed the subtitle in an image with the title and logo, and specify the image using one of the image properties.
  - **titleColor:** A valid HTML color to use for the title of the banner. For example, **red** or **#FF0000**.
  - **subtitleColor:** A valid HTML color to use for the subtitle of the banner. For example, **green** or **#00FF00**.
  - **backgroundColor:** A valid HTML color to use as the background of the banner. For example, **blue** or **#0000FF**.
  - **backgroundImage:** The URL to the banner's background image, if you want a background image. Use this property to add texture to the background color.
  - **leftImage:** The URL to the image that forms the left side of the banner. This property is often used for the image containing the logo, title, and subtitle. The maximum image height and width is 60px by 500px.
  - **leftImageDescription:** The alternative (`alt`) text for the Left Image. The alt text is used by screen readers. It is also used if the image cannot be displayed.
  - **rightImage:** The URL to the image that forms the right side of the banner.
  - **rightImageDescription:** The alternative (`alt`) text for the Right Image. The alt text is used by screen readers. It is also used if the image cannot be displayed.
  - **height:** The height of the banner, in pixels. The default banner is 60 pixels high.

## 15.8 BarcodeScanner Module

The BarcodeScanner Module provides barcode and QR code scanning functionality when the viewer is run in the Geocortex Mobile App Framework. This allows a user to edit a feature by scanning a barcode or QR code that matches a configurable field name. If the scanned value does not match any existing field name, the user can use the scanned value to either create a new feature with the user's current position, or select a feature to assign the scanned value.

To provide this functionality, use the `CreateOrEditFeatureFromBarcodeScan` command, and supply a command parameter object or JSON string with two properties:

- **featureServiceID:** The ID of the feature service for which to create or edit a feature.
- **scanResultFieldName:** The field name for which to set when creating a new feature, or search to edit an existing feature.

For example, `{"featureServiceID": 1, "scanResultFieldName": "MY_FIELD_NAME"}`.

## Configuration Properties

### Module

- **htmlScannerView:** The view ID for scanning barcodes and QR codes. The default is `BarcodeScannerView`.

### Views

- **BarcodeScannerView:**
  - **jsqrcodeSource:** The combined source files of [jsqrcode](#) (JavaScript QR Code Reader). This script allows you to scan QR codes.
  - **jobDecoderWorkerSource:** The `DecoderWorker.js` file from [JOB](#) (JavaScript-Only Barcode Reader). This script allows you to scan barcodes.

### View Models

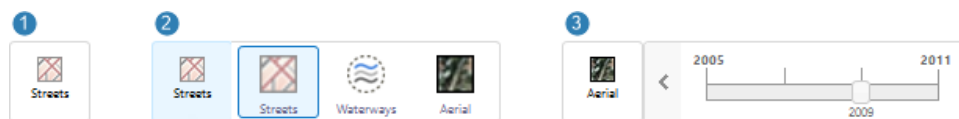
- **BarcodeScannerViewModel:** None

### See also...

[Geocortex SDK for HTML5 API Reference on page 422](#)

## 15.9 Basemap Module

The Basemap Module implements the Basemap Switcher and basemap transparency sliders. The Basemap Switcher provides a quick way for users to make a single basemap visible. Transparency sliders allows users to transition the map smoothly from one map service to the next. See "Basemaps" in the *Geocortex Essentials Administrator Guide* for instructions on configuring basemaps and transparency slider groups.



### Examples of the Basemap Switcher closed (1), open (2) and with a transparency slider (3)

In the Desktop and Tablet interfaces, the Basemap Switcher is on the map. When the user clicks the Switcher, a list of the available basemaps opens. The user clicks a basemap to make it visible. If the basemap contains a transparency slider group, the slider opens. The user drags the slider to transition between the map services in the slider group.

In the Handheld interface, switching basemaps is an option in the I Want To menu. When the user taps the menu option, the list of available basemaps opens so the user can change the basemap.

The Desktop and Tablet interfaces use `BasemapView`. `BasemapView` groups `BasemapSwitcherButtonView` (the Switcher) and `BasemapSliderView` (the Slider). If you move `BasemapView` to a different region, the Switcher and Slider move with it, provided `BasemapSwitcherButtonView` and `BasemapSliderView` are in `BasemapRegion`. `BasemapRegion` inherits its region from `BasemapView`.

In the Desktop and Tablet interfaces, the `BasemapsListController` widget controls the number of rows and columns to show in the Basemap Switcher when the Switcher is open.

## Configuration Properties

### Module

- **None**

### Views

- **BasemapView:** None
- **BasemapSwitcherButtonView:**
  - **hideIfNoBasemapsAvailable:** If this property is set to `true` and there are fewer than two basemaps available, the Basemap Switcher does not display. The default is `true`.
- **BasemapSwitcherView:** None
- **BasemapSliderView:** None

### View Models

- **BasemapSwitcherViewModel:** None

### Widgets

- **BasemapsListController:**
  - **grid:** The basemaps and basemap groups are presented in a grid with the following dimensions:
    - **numberOfRows:** The maximum number of rows to display in the grid. The HTML5 Viewer will not display more than 5 rows. The default is 1.
    - **numberOfColumns:** The maximum number of columns to display in the grid. The HTML5 Viewer will not display more than 5 columns. The default is 4.

The grid that you configure must be able to accommodate at least 2 items (1x2 or 2x1). If the Switcher does not need the full grid to display the basemaps, it shrinks to the appropriate size. If the Switcher contains more basemaps than specified in the widget, the user can scroll the list.

## 15.10 Bookmarks Module



This module can be configured using Manager. See [Configure Map Widgets on page 90](#) for instructions.

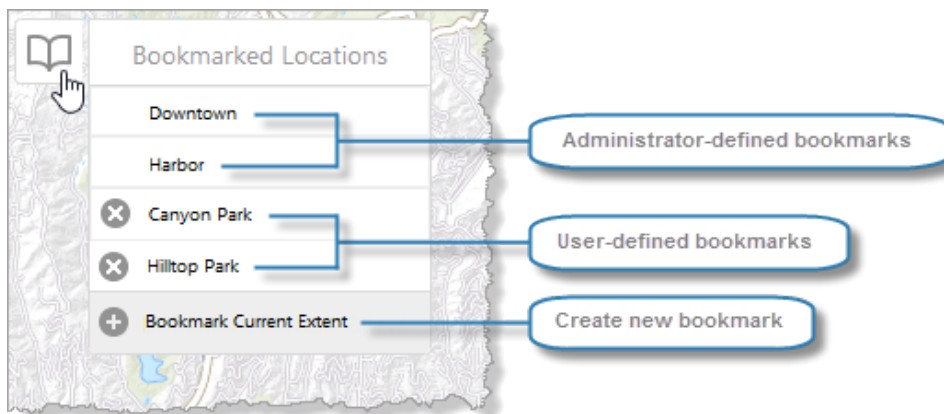
A bookmark is a shortcut that enables end users to jump the map to a particular map extent. The Bookmarks Module implements bookmarks in the HTML5 Viewer.

If there are particular bookmarks that you want to make available to end users, you can configure them in Manager. This saves the bookmarks in the site configuration so that all users have access to them. In addition, end users can create their own bookmarks. User-defined bookmarks are only available to the user who created them.

In order for end users to use bookmarks, the Bookmarks feature must be enabled. This is controlled by the `bookmarksEnabled` property. Bookmarks are enabled by default.

In addition, you must provide a way for users to jump to existing bookmarks and add new bookmarks. The HTML5 Viewer provides a tool and a map widget that give users full access to bookmarks. Alternatively, you could add an I Want To menu item or create a hyperlink that runs the `ShowBookmarks` command. The `ShowBookmarks` command only works if the `bookmarksEnabled` property is set to `true`.

The `ShowBookmarks` command opens the Bookmarked Locations menu, shown below. The Bookmarked Locations menu allows users to use existing bookmarks or add new bookmarks.




### Bookmarked Locations menu

In addition to the `ShowBookmarks` command, the HTML5 Viewer has a `ShowAddBookmarks` command, which opens a dialog box to create a bookmark for the current extent. The `Bookmark Current Extent` item in the Bookmarked Locations menu opens this same dialog box. In addition, the default I Want To menu has an item that runs the `ShowAddBookmarks` command.

The `ShowAddBookmarks` command only works if the `bookmarksEnabled` and `showBookmarksButton` properties are set to `true`.

### Bookmarks Map Widget


The `showBookmarksButton` property controls whether the Bookmarks map widget  shows on the map. By default, the Bookmarks widget appears at the left edge of the map, underneath the Zoom widget. The Bookmarks widget runs the `ShowBookmarks` command, which opens the Bookmarked Locations menu.

The Bookmarks map widget only shows on the map if there is at least one (administrator or user) bookmark defined. If there are no administrator-defined bookmarks, then initially the Bookmarks widget does not show. In this case, users can use the I Want To menu item to create bookmarks. As soon as a user has created one bookmark, the Bookmarks widget shows on the map.

If you define one or more bookmarks in the site, then the Bookmarks map widget is guaranteed to show. In this case, you might want to remove the item from the I Want To menu.

You can control the order of the map widgets in the top-left corner of the map, including the Bookmarks widget. See [Navigation Module on page 285](#) for information.

## Bookmarks Tool

The Bookmarks tool  runs the `ShowBookmarks` command by default. The `ShowBookmarks` command opens the Bookmarked Locations menu. If the Bookmarks map widget is enabled, the menu opens beside the widget. If the map widget is disabled, then the menu opens in a modal window.

## Configuration Properties

### Module

None

### Views

- **BookmarksView:** None

### View Models

- **BookmarksViewModel:**
  - **bookmarksEnabled:** When set to `true`, the Bookmarks feature is enabled—you can add the Bookmarks map widget and tool to the viewer, so users can jump to bookmarks and define new bookmarks. By default, the `bookmarksEnabled` property is `true`.
  - **showBookmarksButton:** When set to `true`, the Bookmarks map widget appears on the map, provided `bookmarksEnabled` is `true` and there is at least one bookmark defined. When `showBookmarksButton` is `true`, the viewer hides the Bookmarks map widget if there are no bookmarks defined. In this case, users can use the **Bookmark current map extent** item in the I Want To menu to create bookmarks. By default, the `showBookmarksButton` property is `false`.

### See also...

[Geocortex SDK for HTML5 API Reference on page 422](#)

[Configure Map Widgets on page 90](#)

## 15.11 Browser Module



This module can be configured using Manager. For instructions, see [Change the Look and Feel on page 74](#).

The Browser Module displays the title that appears in the browser's title bar or tab.

### Configuration Properties

#### Module

- **title:** The title to display in the browser's title bar or tab. The default language key for this property is `@language-browser-title`. For information on working with language keys, see [About User Interface Text on page 64](#).

#### Views

The Browser Module does not have any views.

#### View Models

The Browser Module does not have any view models.

## 15.12 Buffer Module

The Buffer Module implements buffering functionality in the viewer. Buffering allows the surrounding area of interest to be included in an operation, such as the Identify operation. For example, you can identify features within a kilometer radius of a point you click on the map.

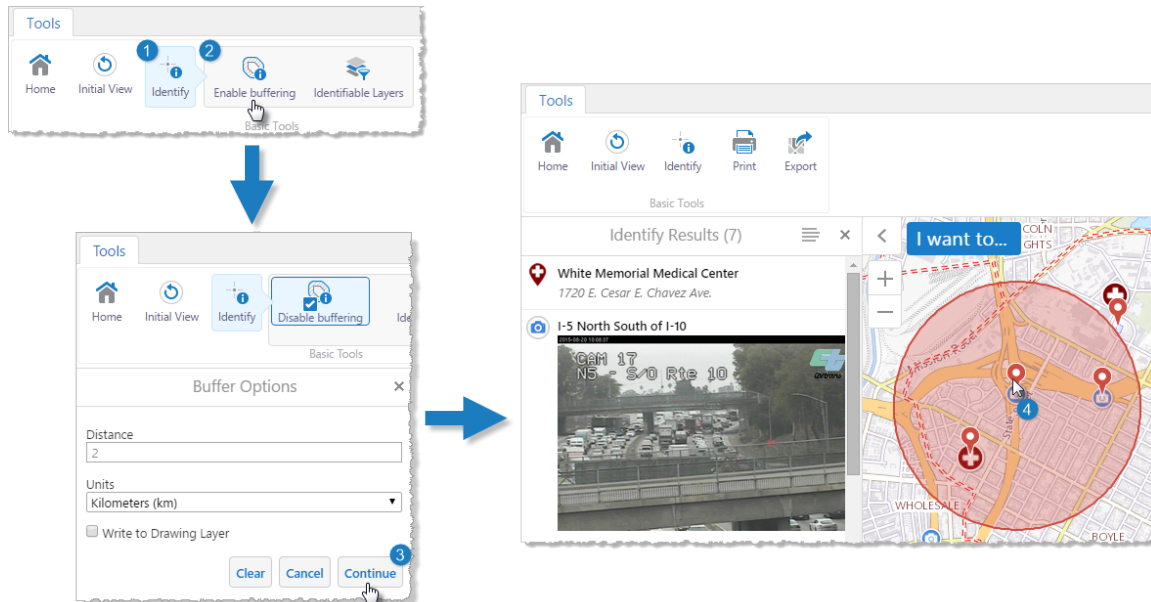
If the Write to Drawing Layer setting is enabled, buffering persists on the map. This allows users to edit and export buffers as drawings.

### Enable Buffering


You can enable buffering in two ways:

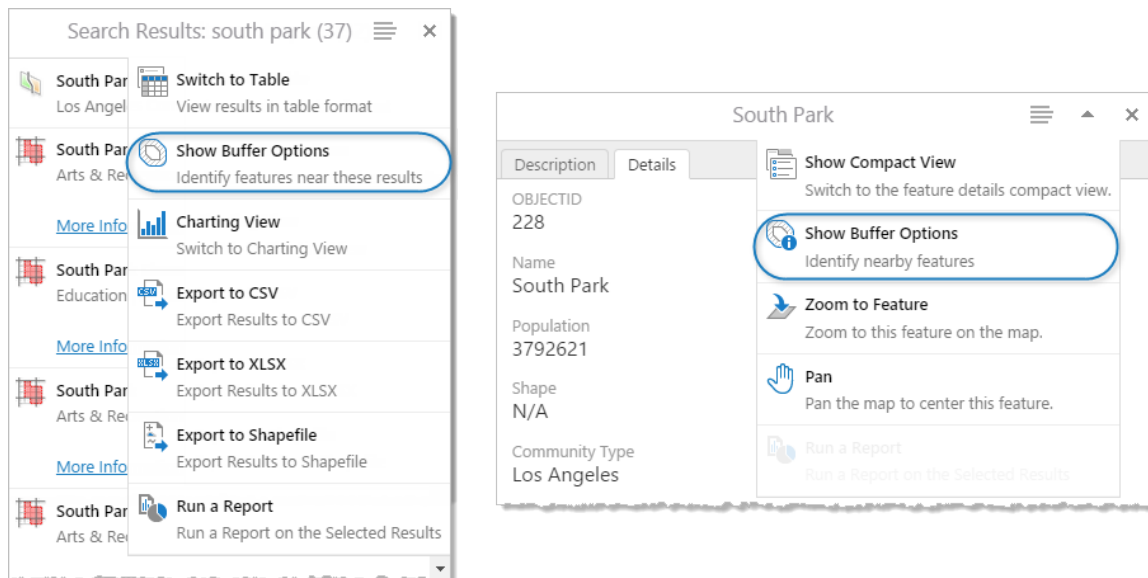


- Activate the Identify multitool and select the **Enable Buffering** tool. The Buffer Options panel is displayed.



The user clicks Identify <sup>1</sup>, Enable Buffering <sup>2</sup>, selects buffer options <sup>3</sup>, and clicks the map to identify the surrounding area <sup>4</sup>

- When the Results List is the active panel, select the **Show Buffer Options** action from the  Panel Actions Menu.



Identify features near those in the Results List (left); identify features near a particular feature

## Buffer Options

From the Buffer Options panel, you can configure the scope of buffers that the user creates:

- **Distance:** The number of units that the buffer extends to from the selected feature outward. You can set the unit of measurement using the Units configuration setting below the Distance setting.
- **Units:** The unit of measurement that the buffer should use to measure distance. You can set the unit of measurement to Feet (ft), Yards (yd), Meters (m), Kilometers (km), Miles (mi), or Nautical Miles (NM).
- **Write to Drawing Layer:** When the Write to Drawing Layer checkbox is selected, the buffer is added to the map as markup and persists on the map.

## Configuration Properties

### Module

- **bufferProjectionWkid:** The well-known ID (WKID) of projection for the Buffer module to use, or an empty string to disable projection. The default is an empty string.
- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **BufferOptionsDismissedBehavior:** A behavior that runs an array of commands when the user dismisses the Buffer Options panel. By default, this includes the command: `CloseDataFrame`.
  - **BufferingErrorBehavior:** A behavior that runs an array of commands when a buffering error occurs. By default, this includes the command: `OpenDataFrame`.
- **numFeaturesWarningThreshold:** The number of features at which the viewer will confirm whether the user wants to proceed with a buffering operation. The purpose of confirmation is to warn the user the operation may take a long time. By default, this property is omitted. If omitted, the default is 250.

### Views

- **BufferOptionsView:**
  - **targetCommands:** An array of commands to which to apply buffering. By default, the commands are: `Identify`, `IdentifyBufferedGeometry`, `IdentifyBufferedFeature`, `IdentifyBufferedFeatureSetCollection` and `IdentifyBufferedFeatureSet`.

### View Models

- **BufferOptionsViewModel:**
  - **addBufferToMarkupLayerByDefault:** If this property is set to `true`, any buffers added by the user are added to the map as markup by default. If this property is absent, it defaults to `false`.
  - **bufferUnits:** An array of distance units to be available for buffering. Possible values include: `feet`, `yard`, `meter`, `kilometer`, `mile`, `nauticalmile`. By default, all units are included except for `yard`.
  - **defaultBufferUnit:** The default unit of distance to use for buffering. The default is `kilometer`.
  - **defaultBufferDistance:** The distance to buffer by default. The default is `0`.

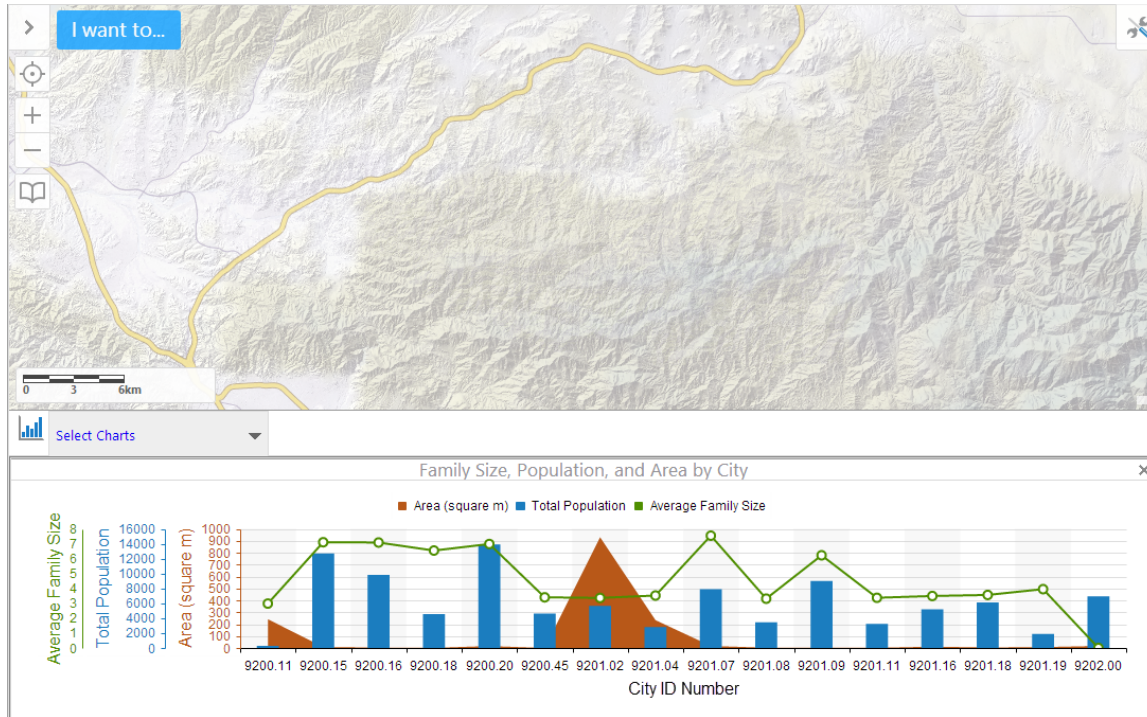
**See Also...**

[Configure Tool Behavior](#) on page 121

[Identify Module](#) on page 193

## 15.13 Charting Module

The Charting Module implements the ability to display charts of a feature layer's data in HTML5 viewers. The data can be from the layer's fields or from a data link that is configured for the layer.

**Chart example**

Charts are configured in Manager. For more information on how to configure charts, refer to the *Geocortex Essentials Administrator Guide*.



It is also possible to display charts for a single feature. For more information, see [FeatureDetails Module](#) on page 168.

## Configuration Properties



The Charting Module requires the Charting library as well as the Mapping.Charting library.

## Module

- **infrastructureLibraryId**: The ID of the infrastructure library. By default, this is `Charting`.
- **adapters**: An array of chart point adapters. Chart point adapters have the following properties:
  - **type**: The fully-qualified type of the chart point adapter. For example, `geocortex.essentialsHtmlViewer.mapping.modules.charting.FeatureChartPointAdapter` or `geocortex.essentialsHtmlViewer.mapping.modules.charting.DataLinkChartPointAdapter`.
  - **source**: The source of the chart data. For example, `Field` or `DataLink`.
  - **configuration**: An object containing chart point adapter configuration. By default, this is empty.
- **behaviors**: An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **ChartPointMouseHoverBeginBehavior**: A behavior that runs an array of commands when the user hovers the mouse over a chart point. By default, this includes two commands: `ClearChartHighlights` and `HighlightChartFeatureSet`.
  - **ChartPointMouseHoverEndBehavior**: A behavior that runs an array of commands when the user stops hovering the mouse over a chart point. By default, this includes the command: `ClearChartHighlights`.
  - **ChartPointMouseDownBehavior**: A behavior that runs an array of commands when the user clicks a chart point. By default, this includes two commands: `ShowMap` and `RunChartFeatureActions`.



You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.



You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.



Adding a new behavior is only recommended for experienced developers.

## Views

- **ChartingView**: None

## View Models

- **ChartingViewModel:**
  - **mobileMode:** When set to `true`, the user can only select one chart at a time in the Handheld interface. The default is `false`. This setting does not affect the Desktop and Tablet interfaces.
  - **chartingEnabled:** To enable charting, set to `true`; otherwise, set to `false`. The default is `true`.
  - **chartConfiguration:** A chart configuration object with the following properties:
    - **animationsEnabled:** To enable chart animations, set to `true`; otherwise, set to `false`. In the Desktop and Tablet interfaces, the default is `true`. In the Handheld interface, the default is `false`.
    - **gradientsEnabled:** To enable chart gradients, set to `true`; otherwise, set to `false`. The default is `false`.
    - **interactiveLegendEnabled:** To enable interactivity with legend items, set to `true`; otherwise, set to `false`. The default is `false`. When enabled, clicking legend items of pie charts will enable or disable slices of the pie, and clicking legend items of linear charts will toggle the display of corresponding series.
    - **pieStartAngle:** The angle at which to start pie charts in degrees. The default is `180`.
    - **renderAs:** The format in which to render charts. The default is `svg`. Possible values include:
      - `svg` renders the chart as an inline SVG document, if available.
      - `vml` renders the chart as VML, if available.
      - `canvas` renders the chart as a Canvas element, if available.
    - **chartWidth:** The width of charts in pixels. The default is `450`.
    - **chartHeight:** The height of charts in pixels. The default is `220`.
  - **infrastructureLibraryId:** The ID of the infrastructure library. By default, this is `Charting`.
  - **containerRegionName:** The name of the container region in which to host charts. The default is `ChartingRegion`.
  - **containerRegionType:** The type of the container region in which to host charts. The default is `geocortex.framework.ui.DivStackRegionAdapter`.
  - **showXButton:** To display a button to close charts, set to `true`; otherwise, set to `false`. The default is `true`.
  - **defaultViewIcon:** The path to an the default charting view icon. The default is `Resources/Images/Icons/Toolbar/search-24.png`.
  - **headerIsVisible:** To display the charting header, set to `true`; otherwise, set to `false`. The default is `true`.
  - **showHeaderForStandaloneViews:** To display the charting headers for standalone views, set to `true`; otherwise, set to `false`. The default is `false`.
  - **backButtonOnRootView:** To display a back button on the root view, set to `true`; otherwise, set to `false`. The default is `false`.

- **showBackButtonAsX**: To display the back button as an X, set to `true`; otherwise, set to `false`. The default is `true`.
- **showMaximizeButton**: To display a maximize button for the container, set to `true`; otherwise, set to `false`. The default is `true`.
- **showHostedViews**: To display views hosted in this view model, set to `true`; otherwise, set to `false`. The default is `false`.
- **resizeY**: To allow the container to be vertically resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **ordering**: A mapping of views and their order ranking, starting with 0. A view with the rank of 0 is the root view of a particular view container. By default, this is empty.

## Example: Add a Command with a Parameter to a Behavior

The following example demonstrates adding a new command with a parameter to the behavior, `ChartPointMouseHoverBeginBehavior`.

### To add a command with a parameter to a behavior:

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST  
Elements\Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. In the `Charting` module section, find the `behaviors` property. Locate the behavior you want to edit. For example, `ChartPointMouseHoverBeginBehavior`.

```

{
  "moduleName": "Charting",
  ...
  "configuration": {
    ...
    "behaviors": [
      {
        "name": "ChartPointMouseHoverBeginBehavior",
        "event": "ChartPointMouseHoverBeginEvent",

        "commands": [
          "ClearChartHighlights",
          "HighlightChartFeatureSet"
        ]
      },
      ...
    ]
  },
  ...
}

```

The behavior executes two commands: `ClearChartHighlights` and `HighlightChartFeatureSet`.

4. Consult the Geocortex SDK for HTML5 API Reference to determine the type of parameter that is associated with these commands. While `ClearChartHighlights` does not have any parameter, `HighlightChartFeatureSet` has a parameter of type, `geocortex.essentialHtmlViewer.mapping.infrastructure.FeatureSet`.
5. Find a command that you want to add to the behavior in the Geocortex SDK for HTML5 API Reference that has the same parameter type. For example, `ZoomToFeatures`.
6. Add the desired command to the list of commands, separated by a comma. For example:

```

"commands": [
  "ClearChartHighlights",
  "HighlightChartFeatureSet",
  "ZoomToFeatures"
]

```

7. Save the file.

#### See Also...

[FeatureDetails Module on page 168](#)

## 15.14 ClusterLayers Module

The ClusterLayers Module works with a number of other modules to implement clustering. For information about configuring clustering for a layer, refer to the *Geocortex Essentials Administrator Guide*.

Specifically, the ClusterLayers Module implements the `visualizationProvider` for clustering and the commands and events that work with clusters. It also tracks the cluster layers that are enabled. A cluster layer is a graphics layers that sits on top of a feature layer that has clustering enabled. When the user turns on clustering or the viewer loads a layer that has clustering turned on, a cluster layer is created for that feature layer. The ClusterLayers Module uses the data from the feature layer to calculate the clusters and renders the cluster symbols on the cluster layer. When the user turns off clustering, the cluster layer is removed.

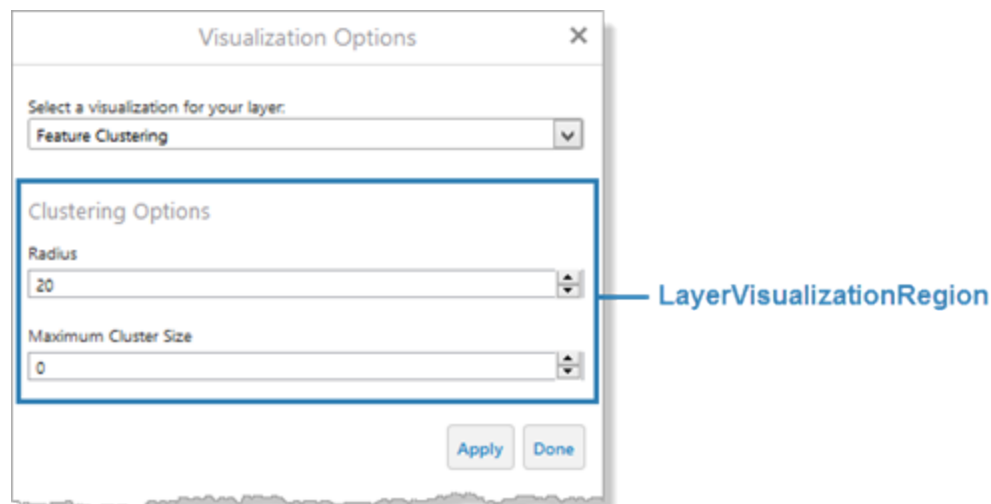
The ClusterLayers Module provides two commands and two events, which you can use in hyperlinks and workflows. The `AddClusterLayer` command creates a cluster layer for the specified feature layer, calculates the clusters, and renders the cluster symbols. `ClusterLayerAddedEvent` is raised when a cluster layer is added to the map. The `RemoveClusterLayer` command removes the cluster layer for the specified feature layer.

`ClusterLayerRemovedEvent` is raised when a cluster layer is removed. At most one visualization can be active at a time. For more information about commands and events, refer to the Geocortex SDK for HTML5 API Reference.

Instructions for accessing the API Reference are [here](#).

The ClusterLayers Module works with the Menu Module, Visualization Module, and Results Module. The [Menu Module](#) has a `LayerActions` menu item, **Turn on/off layer visualizations**, that open the Visualization Options panel, where the user can change visualizations and their settings. The Visualization Options panel is implemented by the [Visualization Module](#). The [Results Module](#) has a `resultMappings` item called `ClusterFeatures` that lists the commands to run when the user clicks a cluster symbol.

The ClusterLayers Module has one view, `ClusterLayerView`. `ClusterLayerView` presents the clustering settings that end users can configure. By default, `ClusterLayerView` is hosted in the `LayerVisualizationRegion` container region, which is reference by the Visualization Module's `VisualizationViewModel`.





---

## Configuration Properties

### Module

- **hideClusters:** Choose to show or hide other clusters when a cluster is selected on the map. Configurable options include `none`, which does not hide other clusters, or `unselected`, which hides other clusters. The default value is `none`.

### Views

- **ClusterLayerView:** None

### View Models

- **ClusterLayerViewModel:** None

### See Also...

[Menu Module on page 282](#)

[Results Module on page 328](#)

[Visualization Module on page 390](#)

## 15.15 Collaboration Module

The Collaboration Module allows users to share text messages, images, and markup in real time, directly in their mapping applications. For further information, see "Configure Collaboration" in the *Geocortex Essentials Administrator Guide*.

## Configuration Properties

### Module

None

### Views

- **CollaborationView:** None
- **CollaborationMarkupStyleSelectorView:** None
- **EventCopyView:** None
- **EditEventView:** None
- **AfterActionView:** None

## View Models

- **EventEditViewModel:**
  - **characterLimit:** The maximum number characters permitted in the chat field when editing an event's message. The default is 1000 characters.
  - **editableTypes:** The types of events that can have their message edited.
- **AfterActionViewModel:**
  - **numEventsToFetch:** The number of events to retrieve when opening a room or when a users clicks the Get previous messages button. The default is 25 events.
  - **previewViewSize:** The dimensions for the preview of an image event in the chat field before it is uploaded to the room. The default values are 50 pixels high and 50 pixels wide.
  - **thumbnailViewSize:** The dimensions for the thumbnail of an image event in the after it is uploaded to the room. The default values are 200 pixels high and 200 pixels wide.
  - **icons:** The icon that is placed on the map to represent an image event. The default values are 24 pixels wide and 24 pixels high, with offsetY and offsetX set to 0.
    - **photoIcon:** The location of the icon for an image event when it is sent to the room. the location of the icon is "Resources/Images/Pushpins/map-marker-image-24.png"
- **CollaborationViewModel:**
  - **characterLimit:** The maximum number of characters permitted in a chat field. The default value is 1000 characters.
  - **numEventsToFetch:** The number of events to retrieve when opening a room or when a users clicks the Get previous messages button. The default is 25 events.
  - **initialRooms:** A list of rooms to automatically join when they become available. This setting is on the Collaboration configuration page in the Viewer settings in Manager.
  - **previewViewSize:** The dimensions for the preview of an image event in the chat field before it is sent to the room. The default values are 50 pixels high and 50 pixels wide.
  - **thumbnailViewSize:** The dimensions for the thumbnail of an image event in the after it is uploaded to the room. The default values are 200 pixels high and 200 pixels wide.
  - **icons:** The icon that is placed on the map to represent an image event. The default values are 24 pixels wide and 24 pixels high, with offsetY and offsetX set to 0.
    - **photoIcon:** The location of the icon for an image event when it is sent to the room. The location of the icon is "Resources/Images/Pushpins/map-marker-image-24.png"
    - **unpostedPhotoIcon:** The location of the icon for an image event before it is sent to the room. The location of the icon is "Resources/Images/Pushpins/map-marker-image-24.png"

## 15.16 CompactToolbar Module

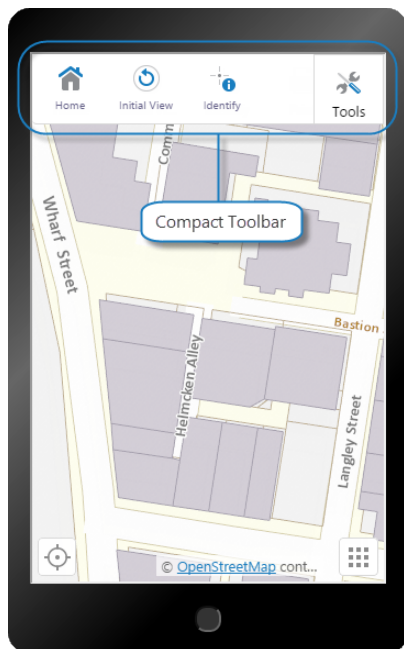


The toolbar can be configured using Manager. See [Configure the Toolbar on page 108](#) for instructions.

The CompactToolbar Module implements the Compact Toolbar typically found in the Handheld interface. When activated, the Compact Toolbar appears at the top of the map.



As of version 2.4, the Handheld interface uses the Compact Toolbar by default, while the Desktop and Tablet interfaces use the Tabbed Toolbar by default. The Tabbed Toolbar is implemented by the [TabbedToolbar Module](#). It is possible to use the Compact Toolbar in the Desktop and Tablet interfaces.




The Handheld interface, showing the Compact Toolbar

## Compact Toolbar

The Compact Toolbar is made up of a single toolbar group, containing buttons, tools or multitools (also known as flyouts). Buttons immediately run commands that do not require input from the user. Tools run commands that operate on a geometry that the user draws; when the user clicks a tool, the tool must wait for the user to draw the geometry before running the command. Multitools act as menus of various related tools or buttons.

In the Handheld interface, the Compact Toolbar has three views, all of which use the `CompactToolbarViewModel`:

- **Compact Toolbar:** The `CompactToolbarView` is used for the Compact Toolbar that, when activated, appears at the top of the screen in the Handheld interface, or below the banner in the Desktop and Tablet interfaces.
- **Compact Toolbar Flyout:** The `CompactToolbarFlyoutView` is used to display the content of multitools.
- **Compact Toolbar Button:** (Handheld interface only) The `CompactToolbarButtonView` is used for the toolbar icon  that the user clicks to open the Compact Toolbar. The toolbar icon displays in the `HeaderRegion` in the Handheld interface.

## Configuration Properties

### Module

- **isEnabled:** To enable the Compact Toolbar, set to `true`; otherwise, set to `false`. The default is `false`.
- **transientElements:** An array of elements that define context-sensitive toolbars, each of which are associated with a [state](#), widget, region and a set of toolbar items.



As of HTML5 Viewer 2.5, each element must be associated with an application [state](#) to create context-sensitive toolbars.

- **stateName:** The name of the application state that triggers the context-sensitive toolbar.



For a complete list of states, see the [State Reference on page 452](#).

- **widgetId:** The ID of the widget.
- **region:** The name of the region to use. For the Compact Toolbar, this is typically set to `CompactToolbarTransientRegion`.
- **items:** An array of toolbar items, each of which is either a button or toggle button.

### Properties of Buttons

- **id:** A unique ID for this button.
- **type:** The type is `button`.
- **iconUri:** The URI for the icon that you want to appear on the button. The image must be an appropriate size to fit on the button. Valid file formats are PNG, BMP, JPG, and JPEG.
- **command:** The command that the button runs when the user clicks the button. For information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters. For information on a particular command's parameter, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **hideOnDisable:** If this property is set to `true` and the button's command cannot run under the current configuration or run-time conditions, the button does not show in the toolbar. If `hideOnDisable` is `false` and the button's command cannot run, the button shows in the toolbar, but it is grayed out.
- **name:** The name that you want to appear on the button. You can use a text key or the literal text. For example, `@language-toolbar-home-sub` or `Home`.

- **tooltip:** The text for the tool tip that opens when the user positions the pointer over the button. You can use a text key or the literal text.  
For example, **@language-toolbar-navigation-home-tooltip** or **Returns to introductory page**.

## Properties of Toggle Buttons

- **id:** A unique ID for this `toggleButton`.
- **type:** The type is `toggleButton`.
- **toggleStateName:** (Optional) The name of the toggle [state](#) that this toggle button affects.
- **toggleOn:** Configures the toggle-on button, which turns the toggle button on:
  - **name:** The name that you want to appear on the toggle-on button. You can use a text key or the literal text.  
For example, **@language-toolbar-home-sub** or **Home**.
  - **tooltip:** The text for the tool tip that opens when the user positions the pointer over the toggle-on button. You can use a text key or the literal text.  
For example, **@language-toolbar-navigation-home-tooltip** or **Returns to introductory page**.
  - **iconUri:** The URI for the icon that you want to appear on the toggle-on button. The image must be an appropriate size to fit on the toggle-on button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable:** If this property is set to `true` and the toggle-on button's command cannot run under the current configuration or run-time conditions, the toggle-on button does not show in the toolbar.  
If `hideOnDisable` is `false` and the toggle-on button's command cannot run, the toggle-on button shows in the toolbar, but it is grayed out.
  - **command:** The command that the toggle-on button runs when the user clicks the toggle button to turn it on.  
For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).
  - **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).
- **toggleOff:** Configures the toggle-off button, which turns the toggle button off:
  - **name:** The name that you want to appear on the toggle-off button. You can use a text key or the literal text.  
For example, **@language-toolbar-markup-clear** or **Clear Markup**.

- **tooltip:** The text for the tool tip that opens when the user positions the pointer over the toggle-off button. You can use a text key or the literal text.  
For example, **@language-toolbar-markup-clear-tooltip** or **Clear all drawings from the map**.
  - **iconUri:** The URI for the icon that you want to appear on the toggle-off button. The image must be an appropriate size to fit on the toggle-off button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable:** If this property is set to `true` and the toggle-off button's command cannot run under the current configuration or run-time conditions, the toggle-off button does not show in the toolbar.  
If `hideOnDisable` is `false` and the toggle-off button's command cannot run, the toggle-off button shows in the toolbar, but it is grayed out.
  - **command:** The command that the toggle-off button runs when the user clicks the toggle button to turn it off.  
For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).
  - **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).
- **toolbarGroups:** An array of `toolbarGroup` items, containing a single `toolbarGroup` with the ID, `compactToolbar`.



Unlike the Tabbed Toolbar, the Compact Toolbar cannot have multiple tabs or groups. It should only contain a single toolbar group with the ID, `compactToolbar`.

## Properties of Toolbar Group

- **id:** For the Compact Toolbar, this ID should always be `compactToolbar`.
- **type:** The type is `toolbarGroup`.
- **name:** This property is not used by the Compact Toolbar.
- **items:** An array of toolbar items, each of which is either a button, toggle button, tool, or flyout (also known as a multitool).

## Properties of Buttons

- **id:** A unique ID for this button.
- **type:** The type is `button`.
- **iconUri:** The URI for the icon that you want to appear on the button. The image must be an appropriate size to fit on the button. Valid file formats are PNG, BMP, JPG, and JPEG.

- **command:** The command that the button runs when the user clicks the button.  
For information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters. For information on a particular command's parameter, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **hideOnDisable:** If this property is set to `true` and the button's command cannot run under the current configuration or run-time conditions, the button does not show in the toolbar. If `hideOnDisable` is `false` and the button's command cannot run, the button shows in the toolbar, but it is grayed out.
- **name:** The name that you want to appear on the button. You can use a text key or the literal text. For example, `@language-toolbar-home-sub` or `Home`.
- **tooltip:** The text for the tool tip that opens when the user positions the pointer over the button. You can use a text key or the literal text. For example, `@language-toolbar-navigation-home-tooltip` or `Returns to introductory page`.

## Properties of Toggle Buttons

- **id:** A unique ID for this `toggleButton`.
- **type:** The type is `toggleButton`.
- **toggleStateName:** (Optional) The name of the toggle [state](#) that this toggle button affects.
- **toggleOn:** Configures the toggle-on button, which turns the toggle button on:
  - **name:** The name that you want to appear on the toggle-on button. You can use a text key or the literal text.  
For example, `@language-toolbar-home-sub` or `Home`.
  - **tooltip:** The text for the tool tip that opens when the user positions the pointer over the toggle-on button. You can use a text key or the literal text.  
For example, `@language-toolbar-navigation-home-tooltip` or `Returns to introductory page`.
  - **iconUri:** The URI for the icon that you want to appear on the toggle-on button. The image must be an appropriate size to fit on the toggle-on button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable:** If this property is set to `true` and the toggle-on button's command cannot run under the current configuration or run-time conditions, the toggle-on button does not show in the toolbar.  
If `hideOnDisable` is `false` and the toggle-on button's command cannot run, the toggle-on button shows in the toolbar, but it is grayed out.
  - **command:** The command that the toggle-on button runs when the user clicks the toggle button to turn it on.  
For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).

- **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).
- **toggleOff:** Configures the toggle-off button, which turns the toggle button off:
  - **name:** The name that you want to appear on the toggle-off button. You can use a text key or the literal text.  
For example, **@language-toolbar-markup-clear** or **Clear Markup**.
  - **tooltip:** The text for the tool tip that opens when the user positions the pointer over the toggle-off button. You can use a text key or the literal text.  
For example, **@language-toolbar-markup-clear-tooltip** or **Clear all drawings from the map**.
  - **iconUri:** The URI for the icon that you want to appear on the toggle-off button. The image must be an appropriate size to fit on the toggle-off button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable:** If this property is set to `true` and the toggle-off button's command cannot run under the current configuration or run-time conditions, the toggle-off button does not show in the toolbar.  
If `hideOnDisable` is `false` and the toggle-off button's command cannot run, the toggle-off button shows in the toolbar, but it is grayed out.
  - **command:** The command that the toggle-off button runs when the user clicks the toggle button to turn it off.  
For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).
  - **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).

## Properties of Tools

- **id:** A unique ID for this `tool`.
- **type:** The type is `tool`.
- **iconUri:** The URI for the icon that you want to appear on the tool. The image must be an appropriate size to fit on the tool. Valid file formats are PNG, BMP, JPG, and JPEG.
- **command:** The command that the tool runs after the user has drawn the geometry for the command to operate on.  
For information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **drawMode:** The type of geometry the user draws, upon which the tool operates.



- **name:** The name that you want to appear on the tool. You can use a text key or the literal text. For example, `@language-toolbar-tasks-identify` or **Identify**.
- **tooltip:** The text for the tool tip that opens when the user positions the pointer over the tool. You can use a text key or the literal text. For example, `@language-toolbar-identify-point-tooltip` or **Find out about a location on the map**.
- **hideOnDisable:** If this property is set to `true` and the tool's command cannot run, the tool does not show in the toolbar. If `hideOnDisable` is `false` and the tool's command cannot run, the tool shows in the toolbar, but it is grayed out.
- **isSticky:** When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time. If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
- **statusText:** The status message to display when the tool is activated, often containing instructions for the user. You can use a text key or the literal text. For example, `@language-toolbar-identify-point-desc` or **Click or tap a location on the map to learn what's there**.

### Properties of Flyouts

- **id:** A unique ID for this flyout.
  - **type:** The type is `flyout`.
  - **name:** The name that you want to appear on the Multitool. You can use a text key or the literal text. For example, `@language-toolbar-markup-drawing-tools` or **Draw**.
  - **items:** An array of items, each of which is either a button, toggle button, or tool.
  - **layout:** This property is not used.
- **layout:** This property is not used by the Compact Toolbar.

### Views

- **CompactToolbarView:** None
- **CompactToolbarFlyoutView:** None
- **CompactToolbarButtonView:** (Handheld interface only) None

## View Models

- **CompactToolBarViewModel:**

- **toggleCommandDisablesView:** (Handheld interface only) To deactivate the `CompactToolBarView` when the user hides the toolbar, set to `true`; otherwise, set to `false`. The default is `true`.



Do not change this property for the Handheld interface or add this property to the Desktop or Tablet interfaces. Doing so may cause the toolbar to behave unexpectedly or stop working altogether.

- **toolbarVisibleTools:** (Desktop and Tablet interfaces) The number of tools to display at once. We recommend a minimum of 3 and a maximum of 9. If this property is missing or its value is 0, all tools will be displayed. The default is 4.
- **toolbarOpenByDefault:** To open the toolbar when the viewer starts, set to `true`; otherwise, set to `false`. The default is `false`.
- **toolbarGroupRefs:** An array of IDs of the configured groups that you want to appear in the toolbar. In the case of the Compact Toolbar, the only group should be `compactToolBar`. Hide the group by removing its ID from the `toolbarGroupRefs` list. Add the group back by adding its ID back to the `toolbarGroupRefs` list.

- **CompactToolBarTransientViewModel:** None

### See Also...

[About User Interface Text on page 64](#)

[TabbedToolBar Module on page 363](#)

## 15.17 Confirm Module

The Confirm Module creates and displays confirmation dialog boxes. The Confirm Module can display built-in messages, or you can configure custom messages to display.

## Configuration Properties

### Module

- **confirmRegion:** The region in which to display the confirmation message. The default region is `ModalWindowRegion`. For a list of regions, see [Regions on page 444](#).

### Views

- **ConfirmView:** None

### View Models

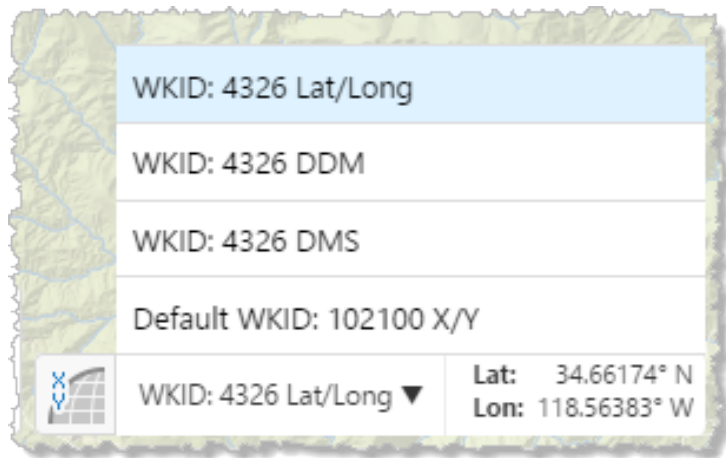
None

## 15.18 Coordinates Module



This module can be configured using Manager. See [Configure Map Widgets on page 90](#) for instructions.

The Coordinates Module displays the map coordinates of the current position of the mouse pointer. Users may select from a variety of coordinate systems and formats. The map coordinates may be opened or closed.



Map coordinates with the coordinate system menu open



The Map Coordinates widget is only available for the Desktop interface.

## Configuration Properties

### Module

- None

### Views

- `CoordinatesView`: None

### View Models

- `CoordinatesViewModel`:
  - `isEnabled`: To enable the coordinates feature, set to `true`; otherwise, set to `false`. The default is `true`.
  - `openByDefault`: To open the coordinates when the viewer starts, set to `true`; otherwise, set to `false`. The default is `false`.

`coordinatesModel`: Sets the coordinates view model. The default value is `MapCoordinatesModel`, which is a view model configured as part of the [Map Module](#).

- **useBasemapCoordinates**: To include the basemap's coordinate system, set to `true`; otherwise, set to `false`. The default is `true`.
- **numDigits**: The number of decimal places to use for coordinates. The default is 5.
- **coordinateSystems**: An array of coordinate systems. Coordinate systems have the following properties:
  - **displayName**: The name to display for this coordinate system.
  - **wkid**: The well-known ID of the coordinate system. This property overrides the `wkt` property.



For a list of WKIDs, see Esri's [Geographic Coordinate Systems](#) or [Projected Coordinate Systems](#) documentation.

- **wkt**: The well-known text of the coordinate system. This property is overridden by the `wkid` property.



For a list of WKTs, see Esri's [Geographic Coordinate Systems](#) or [Projected Coordinate Systems](#) documentation.

- **output**: The units to use for the coordinates. Possible values include:
  - `dms`: Degrees, Minutes, Seconds
  - `ddm`: Degrees, Decimal Minutes
  - `latLon`: Latitude, Longitude
  - `xy`: X, Y (in the units of the coordinate system)

## 15.19 Editing Module

The Editing Module implements editing of features and related features in the viewer.

The Editing Module depends on the [FeatureLayer Module](#), which implements feature layers in the HTML5 viewer. It also works with the [Offline Module](#) to support offline editing. In order for features to be editable, they must belong to a feature layer that has editing turned on. In order to edit related features from ArcGIS Tables, you must also add the tables in Manager. For more information, see "ArcGIS Tables" in the *Geocortex Essentials Administrator Guide*.

The Editing Module has different views for the different parts of the viewer that provide access to feature editing: `MapDataMenuView`, `TemplatePickerView`, `EditorView`, `EditLogView`, `CreateOrEditView`, and `MultiFeatureSelectorView`.

The Editing Module has the following view models: `TemplatePickerViewModel`, `EditLogViewModel`, `CreateOrEditViewModel`, `EditorViewModel`, and `EditingMapDataMenuViewModel`.

### Create Point Features Using Geolocation

As of HTML5 Viewer 2.5, users can create point features by using geolocation. To enable this feature, edit your viewer in Essentials Manager, and in the Toolbar section, ensure both **Create New Feature** and **CreateNewFeatureControlRegion** are included in your configured toolbar. You can configure geolocation settings with the Geolocate module. For information see [Geolocate Module on page 184](#).

When the user selects a point feature template, a menu appears that offers geolocation. If the geolocation result is not within a configurable accuracy threshold, the user is prompted whether to use the result or select a different location on the map. The accuracy threshold is specified in the context-sensitive toolbar that is associated with the Geolocate module's `FeaturePlacementPointGraphicState` by configuring a `accuracyThreshold` property. You may also specify the number of milliseconds for which to refine the geolocation reading by configuring a `timeLimit` property.

Alternatively, you can specify a geolocation profile by configuring a `profile` property. The geolocation profiles are defined in the `GeolocateViewModel` by the `singleGeolocationProfiles` array. The

**CreateNewFeatureControlRegion** also includes [snapping](#) tools. In the case of the Compact Toolbar, do not include the region.

## Add Attachments to New Features

Users are not able to add an attachment to a feature until after a feature has been created and synced with the server. If an editable layer includes a field for a file attachment, the user must create the feature, then re-select the feature using the Edit Feature action to upload attachments.

## Configuration Properties

### Module

- **behaviors**: An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **EditorFeatureSelectedBehavior**: A behavior that runs an array of commands when a feature is selected while the user is editing features. By default, this includes four commands: `ZoomToFeature`, `SetActiveHighlightLayerDefault`, `ClearHighlights`, and `HighlightFeature`.
  - **EditorRemoveFeatureSelectedBehavior**: A behavior that runs an array of commands when a feature is deselected while the user is editing features, which mainly occurs when the user closes the Feature Attributes panel or edits the geometry of a feature. By default, this includes two commands: `SetActiveHighlightLayerDefault` and `ClearHighlights`.



You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.



You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.



Adding a new behavior is only recommended for experienced developers.

## Views

- **MapDataMenuView:**
  - **menuId:** The ID of the menu that contains options for managing offline data. The default is `MapDataMenu`. The menu is configured in the [Menu Module](#).
- **TemplatePickerView:** None
- **EditorView:** None
- **EditLogView:** None
- **CreateOrEditView:** None
- **MultiFeatureSelectorView:** None

## View Models

- **TemplatePickerViewModel:** None
- **EditLogViewModel:** None
- **CreateOrEditViewModel:**
  - **searchRadiusMeters:** When a scanned QR code or barcode does not match a feature, the user can click the map to edit a feature to assign the scanned value. This property represents the meter radius around the point the user clicks to search for features to edit. For more information, see [BarcodeScanner Module on page 135](#).
  - **tools:** When a scanned QR code or barcode does not match a feature, the user can click the map to edit a feature to assign the scanned value. This property represents the array of tools with which the user can select features to edit. The default includes a single tool, `SelectFeaturesForEditingTool`. For more information, see [BarcodeScanner Module on page 135](#).

The [Tools Module](#) implements the ability to define an array of tools in other modules. Each tool in the array has the following properties:


- **name:** The name of the tool.



If your viewer is going to be available in more than one language, enter the text key that the tool name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **command:** The command that the tool runs.  
For a list of commands, see the [Geocortex SDK for HTML5 API Reference](#).
- **drawMode:** The type of geometry the user draws, upon which the tool operates.
- **isSticky:** When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.  
If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
- **iconUri:** The image that displays beside the tool.

- `statusText`: The status message to display when the tool's input method is via mouse, often containing instructions for the user. You can use a text key or the literal text.
  - `keyboardStatusText`: The status message to display when the tool's input method is via keyboard, often containing keyboard shortcut hints for the user. You can use a text key or the literal text.
- **MultiFeatureSelectorViewModel:**
    - `displayResultPickerTemplateComplexity`: To display only feature labels in the feature selector, set to `simple`; to also display feature descriptions and icons, set to `complex`. The default is `complex`.
  - **EditorViewModel:**
    - `editGeometry`: To allow the geometry of features to be edited, set to `true`; otherwise, set to `false`. The default is `true`.
    - `validateGeometry`: To validate geometry changes when the user saves edited geometry, set to `true`; otherwise, set to `false`. For example, a self-intersecting geometry is invalid. The default is `true`.
    - `tools`: An array of tools for the user to edit the geometry with. The factory configuration has tools to edit points, lines, polylines, freehand polylines, polygons, freehand polygons, circles, ellipses and rectangles. The [Tools Module](#) implements the ability to define an array of tools in other modules. Each tool in the array has the following properties:
      - `name`: The name of the tool.

 If your viewer is going to be available in more than one language, enter the text key that the tool name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.
    - `command`: The command that the tool runs.  
For a list of commands, see the [Geocortex SDK for HTML5 API Reference](#).
    - `drawMode`: The type of geometry the user draws, upon which the tool operates.
    - `isSticky`: When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.  
If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
    - `iconUri`: The image that displays beside the tool.
    - `statusText`: The status message to display when the tool's input method is via mouse, often containing instructions for the user. You can use a text key or the literal text.
    - `keyboardStatusText`: The status message to display when the tool's input method is via keyboard, often containing keyboard shortcut hints for the user. You can use a text key or the literal text.
  - **EditingMapDataMenuViewModel:** None

## Example: Add a Command with a Parameter to a Behavior

The following example demonstrates adding a new command with a parameter to the behavior, `EditorFeatureSelectedBehavior`.

### ► To add a command with a parameter to a behavior:

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. In the `Editing` module section, find the `behaviors` property. Locate the behavior you want to edit. For example, `EditorFeatureSelectedBehavior`.

```
{
  "moduleName": "Editing",
  ...
  "configuration": {
    "behaviors": [
      {
        "name": "EditorFeatureSelectedBehavior",
        "commands": [
          "ZoomToFeature",
          "SetActiveHighlightLayerDefault",
          "ClearHighlights",
          "HighlightFeature"
        ]
      },
      ...
    ]
  },
  ...
}
```

The behavior executes a few commands, some that include a parameter; one such example is `ZoomToFeature`.

4. Consult the Geocortex SDK for HTML5 API Reference to determine the type of parameter that is associated with these commands. `ZoomToFeature` has a parameter of type, `geocortex.essentialsHtmlViewer.mapping.infrastructure.Feature`.
5. Find a command that you want to add to the behavior in the Geocortex SDK for HTML5 API Reference that has the same parameter type. For example, `ShowMapTip`.
6. Add the desired command to the list of commands, separated by a comma. For example:



```
"commands": [  
  "ZoomToFeature",  
  "SetActiveHighlightLayerDefault",  
  "ClearHighlights",  
  "HighlightFeature",  
  "ShowMapTip"  
]
```

7. Save the file.

#### See Also...

[FeatureLayer Module on page 176](#)

[Geocortex SDK for HTML5 API Reference on page 422](#)

[Menu Module on page 282](#)

[Offline Module on page 287](#)

[Tools Module on page 375](#)

[About User Interface Text on page 64](#)

## 15.20 ExportMap Module

The Export Map Module makes it possible for users to export the current map image.

The HTML5 Viewer has an Export tool, which can be added from the list of available tools in Manager. The Export tool calls the `ShowExportMapDialog` command. You can also create menu items, hyperlinks, or workflows that export the map image using the `ShowExportMapDialog` command.



### Export tool

If you want users to be able to include georeference data with the map image you can configure this in Essentials. Including georeference data allows the image to be accurately positioned in other GIS applications. You can also configure the default file format to export the map image to. The user can change the file format. These settings are configured in the site, not in the viewer. For information, see "Configure Map Export" in the *Geocortex Essentials Administrator Guide*.



iOS does not support ZIP files. To use Export features that produce ZIP files in iOS, you must install third-party software to handle ZIP files, such as iZip.

## Configuration Properties

### Module

- None

### Views

- **ExportMapView: None**

### View Models

- **ExportMapViewModel: None**

### See also...

[Geocortex SDK for HTML5 API Reference on page 422](#)

## 15.21 ExportWebMap Module

The ExportWebMap Module allows users to publish the current state of their map as a web map to ArcGIS Online or Portal for ArcGIS Online. Once the web map is exported, it can be viewed in an ArcGIS viewer. The module requires that users be signed in to an organizational ArcGIS Online or Portal for ArcGIS account.

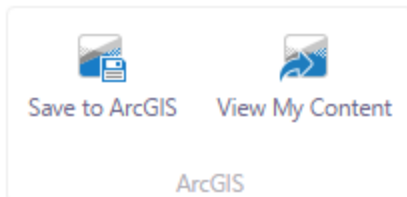
During export, the web map item is assigned an item ID and generates a URL to the new web map item on ArcGIS Online.



Public ArcGIS.com accounts have limited functionality and cannot use the Save to ArcGIS feature.

## Configure the Toolbar

The ExportWebMap Module includes a toolbar group called ArcGIS. The tools in the group are **Save to ArcGIS**, which moves through the publishing process, and **View My Content**, which sends the user to the last published web map item.

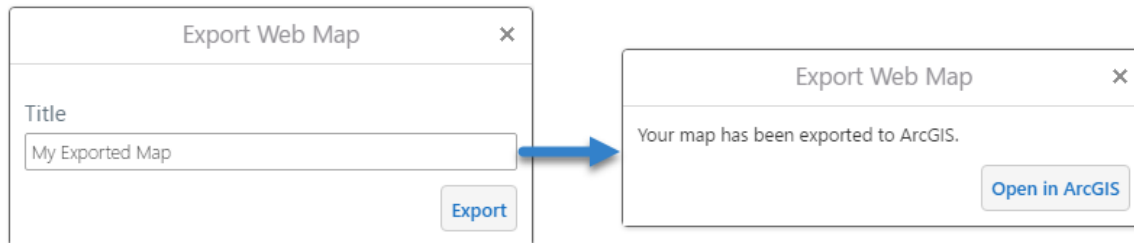


### The ArcGIS tool group

See [Configure the Toolbar on page 108](#) for more information about toolbar configuration.

## Export Web Map

After choosing to export a web map, a dialog box will prompt the user to input a title for the web map. Once the title is submitted, the user can choose to view the details of the exported map on ArcGIS Online or Portal for ArcGIS, or they can choose to return to their map.



### Export Web Map dialog boxes

If the user chooses to return to their map, they can navigate to their ArcGIS Online account's My Content page by choosing the **View My Content** tool if it is included in the configured toolbar.

## View Exported Content on ArcGIS Online or Portal for ArcGIS

If a user is signed in to an ArcGIS Online or Portal for ArcGIS account, they can use the **View My Content** tool to jump to the account's My Content page. Web maps exported from Essentials appear on the My Content page ready for use with ArcGIS Online.



If a web map exported to ArcGIS Online by Essentials references layers that only support unsecured (<http://>) requests, the layer data fails. This is because ArcGIS Online viewers force secure (<https://>) requests. Whenever possible, reference secure, SSL-enabled layers from sources that you trust.

## Export Contents

The exported web map saves the following information from the user's session:

- Map's center point
- Currently selected basemaps
- Layer visibility
- Currently applied layer filters
- Currently selected layer theme
- User-added layers, including layers added dynamically from a layer catalog
- Uploaded files such as CSV files or shapefiles
- Markup (graphics and text) from measurement and drawing tools
- List of saved filters and queries
- Current feature set, and where it is displayed (Results List or Results Table)
- Current feature details, and how they are displayed (compact or expanded)
- Heat maps
- Highlights
- Pushpins
- Plotted coordinates
- Current map tip
- Current mouse coordinates selection
- Bookmarks

- Current set of selected features

## Commands

The `ExportWebMap` Module includes two commands. The `ShowExportWebMapDialog` starts the export process, and the `OpenPortalMyContentWindow` opens a user's ArcGIS Online or Portal for ArcGIS My Content page in a new browser window.

## Configuration Properties

### Module

- `None`


### Views

- `ExportWebMapView: None`

### View Models

- `ExportWebMapViewModel: None`

## 15.22 FeatureDetails Module

The `FeatureDetails` Module displays information related to a spatial feature. As of version 2.4, the Desktop and Tablet interfaces can show feature details in two modes: `Compact View` and `Expanded View`. In the `Compact View`, feature details are displayed as a list in the sidebar. In the `Expanded View`, feature details are displayed as a table in the bottom panel. The user may switch between the different modes via the `Panel Actions Menu`  at the top-right of the panel. By default, the feature details are displayed in the `Compact View`. The `Handheld` interface does not support the `Expanded View`.

Different types of information are obtained from sources that you configure called "feature details providers". The feature details providers are the sources that provide information about the feature. For example, if you want the feature's attributes to display when the user views the feature details, ensure the configuration for the `Attributes Provider` is present. All feature details providers are configured by default.

The HTML5 Viewer has the following feature details providers:

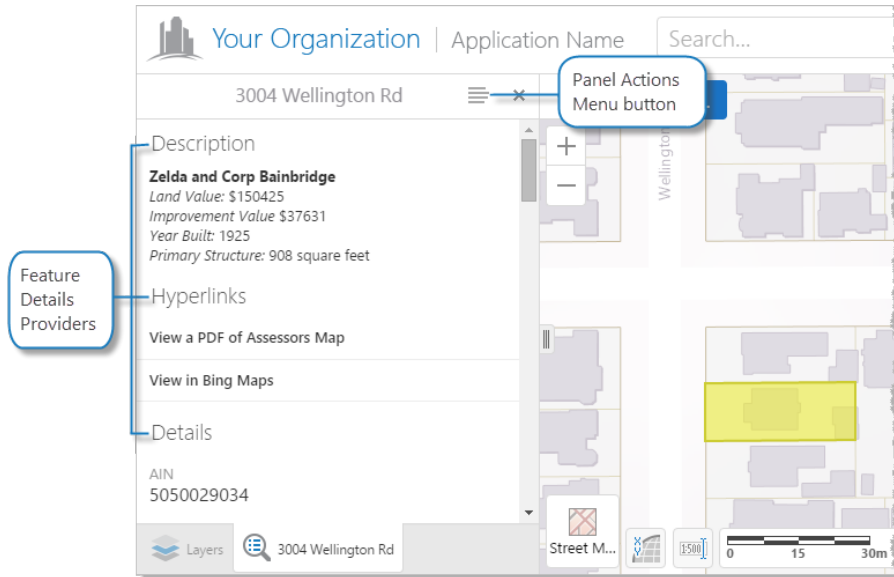
- **Description:** Displays the feature's description or long description.
- **Hyperlinks:** Lists the feature hyperlinks for the current feature.
- **Attributes:** Lists the attributes of the feature, for example, `ObjectId`, `Shape`, `Name`, and so on.
- **Single Feature Charts:** Displays charts about a single feature, as configured in `Manager`.



For more information on how to configure charts, see the *Charts* section in the *Geocortex Essentials Administrator Guide*.

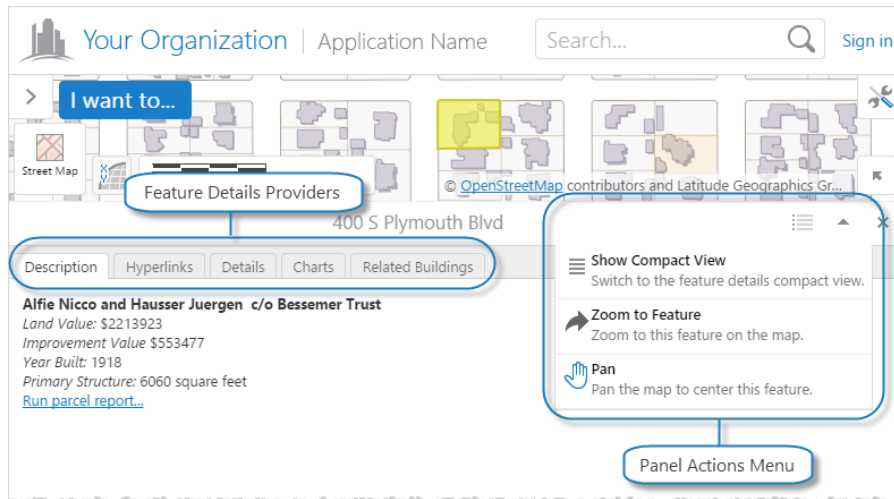
- **Attachments:** Lists the feature's attachments. To delete an attachment, click the **Delete attachment** icon and confirm the deletion when prompted.
- **Related Features:** Lists the relationships that are configured for the current layer.
- **Data Links:** Lists the data linked to by data links that are configured for the current layer.

The Compact View displays the different types of information in a stack, as illustrated below.



**Example of feature detail providers in the Compact View**

The Expanded View displays the different types of information in separate tabs, as illustrated below.



**Example of feature detail providers in the Expanded View**

## Configuration Properties

### Module

- **defaultViewMode:** The default mode with which to view feature details: either `compact` or `expanded`. The default is `compact`. The Handheld interface does not support `expanded`.
- **viewModes:** The different modes available to view feature details:
  - **compact:** The Compact View typically displays the feature details as a list in the sidebar:
    - **viewId:** The ID of the view to use as the Compact View. The default is `FeatureDetailsCompactView`.
    - **defaultProviderTargetRegion:** The default region to use for the feature details provider in the Compact View. The default is `FeatureDetailsCompactViewRegion`.
  - **expanded:** The Expanded View typically displays the feature details as a table in the bottom panel:
    - **viewId:** The ID of the view to use as the Expanded View. The default is `FeatureDetailsExpandedView`.
    - **defaultProviderTargetRegion:** The default region to use for the feature details provider in the Expanded View. The default is `FeatureDetailsBottomRegion`.
- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **FeatureDetailsOpenedBehavior:** A behavior that runs an array of commands when feature details are displayed. By default, this includes four commands: `ZoomToFeature`, `SetActiveHighlightLayerDefault`, `ClearHighlights`, and `HighlightFeature`.
  - **FeatureDetailsClosedBehavior:** A behavior that runs an array of commands when feature details are closed. By default, this includes two commands: `SetActiveHighlightLayerDefault` and `ClearHighlights`.



You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.



You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.



Adding a new behavior is only recommended for experienced developers.

- **providers:** An array of feature details providers. Each provider is a view model that provides information about the feature to the corresponding view, for example, the `RelatedFeaturesViewModel` provides a list of related

features to the `RelatedFeaturesView`.

The configuration properties for the feature details providers are as follows:

- **Description:**



To display any feature description (long or short), set this feature detail provider's `enabled` property to `true`; otherwise, set it to `false`. The default is `false`.

- **longDescription:** To display the feature's long description, set to `true`; otherwise, set to `false`. The default is `false`.

- **Hyperlinks:** None

- **Attributes:** None

- **Single Feature Charts:**

- **infrastructureLibraryId:** The ID of the infrastructure library. By default, this is `Charting`.
- **containerRegionName:** The name of the container region in which to host single feature charts. The default is `SingleFeatureChartsRegion`.
- **chartConfiguration:** A chart configuration object with the following properties:
  - **animationsEnabled:** To enable chart animations, set to `true`; otherwise, set to `false`. The default is `true`.
  - **gradientsEnabled:** To enable chart gradients, set to `true`; otherwise, set to `false`. The default is `false`.
  - **interactiveLegendEnabled:** To enable interactivity with legend items, set to `true`; otherwise, set to `false`. The default is `false`. When enabled, clicking legend items of pie charts will enable or disable slices of the pie, and clicking legend items of linear charts will toggle the display of corresponding series.
  - **pieStartAngle:** The angle at which to start pie charts in degrees. The default is `180`.
  - **renderAs:** The format in which to render charts. The default is `svg`. Possible values include:
    - `svg` renders the chart as an inline SVG document, if available.
    - `vml` renders the chart as VML, if available.
    - `canvas` renders the chart as a Canvas element, if available.

- **Attachments:** None

- **Related Features:** None

- **Data Links:**

- **dataLinkDetailsView:** The region where the view for linked data is hosted. This view is created when the user clicks a linked data item. In the Desktop and Tablet interfaces, the default is

`DataFrameResultsContainerRegion`; in the Handheld interface, the default is `ResultsViewContainerRegion`. For a list of regions, see [Regions on page 444](#).



Each feature detail provider supports an optional `enabled` property, which is on the same level as the `config` property. To enable the feature detail provider, set `enabled` to `true`; otherwise, set it to `false`. The default is `true` by virtue of the feature detail provider's presence, so in most cases, the `enabled` property is omitted.



Each feature detail provider supports an optional `targetRegion` property, which is on the same level as the `config` property. This property overrides the default region where feature details are displayed, as specified by the `defaultProviderTargetRegion` property. By default, if more than one provider targets the same region, tabs appear above the region to allow switching between providers.

To use the `targetRegion` property, you must supply an object with the view mode names as the keys, and the corresponding regions as the values:

```
"providers": [  
  {  
    ...  
    "viewId": "FeatureDescriptionProviderView",  
    "targetRegion": {  
      "compact": "MyCompactRegion",  
      "expanded": "MyExpandedRegion"  
    },  
    ...  
  },  
  ...  
]
```

In the above example, the feature description provider uses the `MyCompactRegion` for the Compact View and the `MyExpandedRegion` for the Expanded View.





Each feature detail provider supports the `markup` property, which is on the same level as the `config` property. This property specifies HTML file to use as the markup for each view mode. To use the `markup` property, you may either supply a string containing the location of the HTML file for all view modes; or an object with the view mode names as the keys, and the corresponding HTML file locations as the values:

```
"providers": [
  {
    ...
    "viewId": "FeatureDescriptionProviderView",
    "markup": {
      "compact": "MyPath/MyCompactView.html",
      "expanded": "MyPath/MyExpandedView.html"
    },
    ...
  },
  ...
]
```

In the above example, the feature description provider uses `MyPath/MyCompactView.html` for the Compact View and the `MyPath/MyExpandedView.html` for the Expanded View.

## Views

- **FeatureDetailsExpandedView:**
  - **onDeactivated:** An array of commands to run when the `FeatureDetailsExpandedView` view is deactivated. By default, one command is run: `ClearDefaultHighlights`.
- **FeatureDetailsCompactView:**
  - **onDeactivated:** An array of commands to run when the `FeatureDetailsCompactView` view is deactivated. By default, one command is run: `ClearDefaultHighlights`.

## View Models

- **FeatureDetailsExpandedViewModel:**
  - **defaultTabViewForLayer:** To set the default tab view for all layers, use the view name as the property and set the value to `default`. To set the default tab view for specific layers, add the view name as the property and set the value to an array of objects for each layer; each object contains the following properties:
    - **mapServiceId:** The ID of the map service that contains the layer for which you want to set the default tab view.
    - **layerId:** The ID of the layer for which you want to set the default tab view.

In the following example, the default tab view is `FeatureChartsProviderView` for all layers except layer 1 of the `Streets` map service and layer 2 of the `Zoning` map service:

```
"defaultTabViewForLayer": {
  "FeatureChartsProviderView": "default",
  "FeatureChartsProviderView": [
    {
      "mapServiceId": "Streets",
      "layerId": "1"
    },
    {
      "mapServiceId": "Zoning",
      "layerId": "2"
    }
  ]
}
```

- **regions:** An array of regions. A region has the following properties:
  - **regionName:** The name of the region.
  - **regionType:** (Optional) The type of region. By default, when multiple feature detail providers target the same region, tabs appear above the region. If you want the views to stack one after another instead, set this property to `geocortex.framework.ui.DivStackRegionAdapter`. By default, this property is omitted.



If you add this property, the CSS class specified by `regionCSS` must be modified accordingly.

- **regionCss:** The Cascading Style Sheet (CSS) class for the region.
- **FeatureDetailsCompactViewModel:**
  - **regions:** An array of regions. A region has the following properties:
    - **regionName:** The name of the region.
    - **regionType:** (Optional) The type of region. By default, when multiple feature detail providers target the same region, tabs appear above the region. If you want the views to stack one after another instead, set this property to `geocortex.framework.ui.DivStackRegionAdapter`. By default, this property is omitted.



If you add this property, the CSS class specified by `regionCSS` must be modified accordingly.

- **regionCss:** The Cascading Style Sheet (CSS) class for the region.

## Example: Add a Command with a Parameter to a Behavior

The following example demonstrates adding a new command with a parameter to the behavior, `FeatureDetailsOpenedBehavior`.

### ► To add a command with a parameter to a behavior:

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Sites\[site]\Viewers\
[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. In the `FeatureDetails` module section, find the `behaviors` property. Locate the behavior you want to edit. For example, `FeatureDetailsOpenedBehavior`.

```
{
  "moduleName": "FeatureDetails",
  ...
  "configuration": {
    ...
    "behaviors": [
      {
        "name": "FeatureDetailsOpenedBehavior",
        "event": "FeatureDetailsCurrentFeatureChanged",
        "commands": [
          "ZoomToFeature",
          "SetActiveHighlightLayerDefault",
          "ClearHighlights",
          "HighlightFeature"
        ]
      },
      ...
    ]
  },
  ...
}
```

The behavior executes a few commands, some that include a parameter; one such example is `ZoomToFeature`.

4. Consult the Geocortex SDK for HTML5 API Reference to determine the type of parameter that is associated with these commands. `ZoomToFeature` has a parameter of type, `geocortex.essentialsHtmlViewer.mapping.infrastructure.Feature`.
5. Find a command that you want to add to the behavior in the Geocortex SDK for HTML5 API Reference that has

the same parameter type. For example, `ShowMapTip`.

6. Add the desired command to the list of commands, separated by a comma. For example:

```
"commands": [  
  "ZoomToFeature",  
  "SetActiveHighlightLayerDefault",  
  "ClearHighlights",  
  "HighlightFeature",  
  "ShowMapTip"  
]
```

7. Save the file.

#### See Also...

[Charting Module on page 143](#)

[Menu Module on page 282](#)

[Regions on page 444](#)

## 15.23 FeatureLayer Module

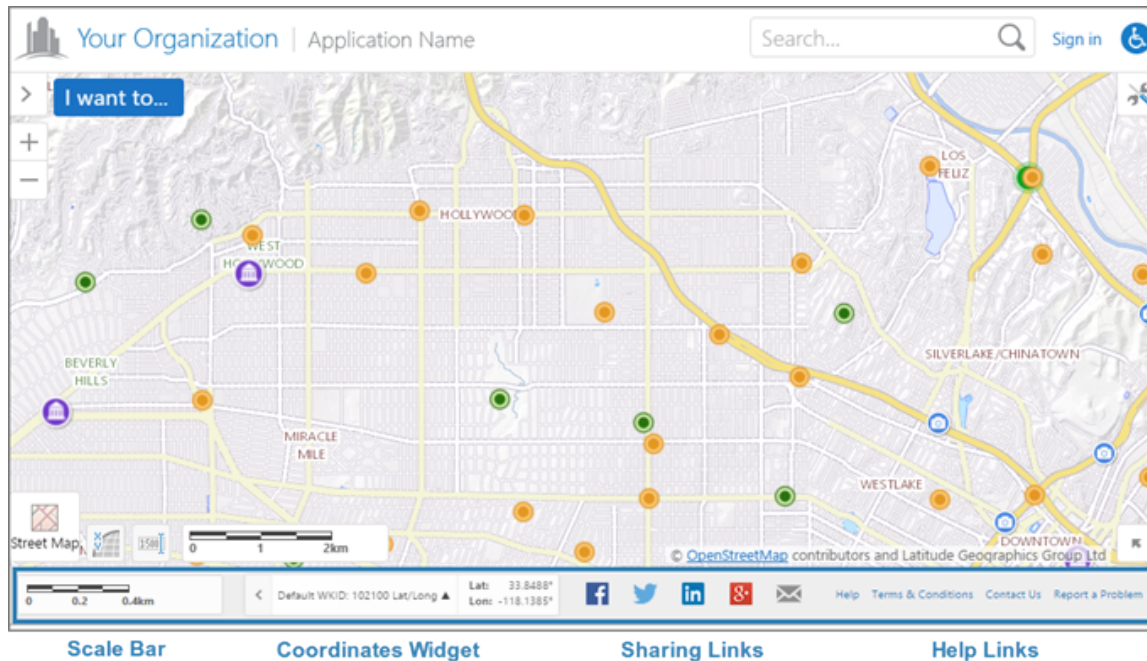
Due to changes to offline functionality in version 2.6, the FeatureLayer Module became unnecessary and no longer exists. Offline functionality requires the Geocortex Mobile App Framework.

## 15.24 Footer Module

The Footer module implements the footer in HTML5 viewers. The footer can appear in the Desktop and Tablet interfaces only—the Handheld interface cannot have a footer. By default, the Footer module's `height` property is set to 0 (zero pixels), which hides the footer. To show the footer, set the `height` to a positive value.

A viewer's footer can contain different types of content:

- **Menus:** A footer menu is an array of items, each of which can optionally run a command. An item can be:
  - **Simple Text:** For example, a copyright statement that is not clickable.
  - **Text Hyperlink:** For example, a link to a help system or website.
  - **Simple Image:** For example, a logo that is not clickable.
  - **Hyperlinked Image:** For example, an email hyperlink or social media hyperlink.
- **Features implemented by other modules:** For example, the scale bar or coordinates widget.



### Example of an HTML5 viewer's footer with a variety of items

To configure text items in a footer menu, configure the item's `text` property, but not its `iconUri` property. To configure image items, configure the item's `iconUri` property, but not its `text` property. You may also want to use different markup—views in the Footer module have two options for the markup that they apply to menu items:

- `MenuView.html`: Provides formatting for simple images, hyperlinked images, and simple text.
- `MenuHyperlinkView.html`: Provides formatting for text hyperlinks. If you use `MenuView.html` instead of `MenuHyperlinkView.html` for text hyperlinks, the hyperlinks will work, but they will not look like hyperlinks.

The footer has its own regions—`FooterRegion`, `LeftFooterRegion`, and `RightFooterRegion`. `FooterRegion` is the entire footer. `LeftFooterRegion` and `RightFooterRegion` lie on top of `FooterRegion`. The contents of the `LeftFooterRegion` are left justified. The contents of the `RightFooterRegion` are right justified.

By default, the Footer module has two views—`FooterView` and `FooterMenuView`. `FooterMenuView` contains a menu of hyperlinks in the `RightFooterRegion` by default.

## Configuration Properties

### Module

- **menus**: An array of menus to display in the footer.
  - **id**: The menu's ID.
  - **description**: A short description of the menu.



If your viewer is going to be available in more than one language, enter the **text key** that the description is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **moduleId**: The ID of the module that the menu belongs to. The `moduleId` is **Footer**.
- **items**: An array of menu items. Menu items have the following properties:
  - **iconUri**: The image to display for the menu item. In order for the icon to show, the view's `markup` property must be set to `Mapping/infrastructure/menus/MenuView.html`.
  - **text**: The text to appear for the menu item. You can use a text key or the literal text. In order for the text to show, the view's `markup` property must be set to `Mapping/infrastructure/menus/MenuHyperlinkView.html`.
  - **description**: A brief description of the menu item to appear below the `text`. You can use a text key or the literal text.
  - **command**: The command to execute when the menu item is selected. A footer menu can run any HTML5 Viewer command. For a list of commands, see "Commands" in the [Geocortex SDK for HTML5 API Reference](#).



If you set the `command` property, do not set the `batch` property.

- **commandParameter**: The parameter value to pass to the command when it runs, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.



If you set the `commandParameter` property, do not set the `batch` property.

- **hideOnDisable**: If this property is set to `true` and the menu item's command cannot run, the menu item does not show in the menu.  
If `hideOnDisable` is `false` and the menu item's command cannot run, the menu item shows in the menu, but it is grayed out.
- **batch**: An array of objects, each with three properties: an executable command, an optional command parameter, and an optional Boolean to abort the execution of further commands if a command fails to execute. This allows multiple commands to be executed in a specified order,

although some commands may trigger asynchronous actions.

The following example attempts to activate the Tabbed Toolbar and, if successful, proceeds to greet the user with an alert:

```
"batch": [  
  {  
    "command": "ActivateView",  
    "commandParameter": "TabbedToolbarView",  
    "abortBatchOnFailure": true  
  },  
  {  
    "command": "Alert",  
    "commandParameter": "Hello!"  
  }  
]
```

If omitted, `abortBatchOnFailure` is `false` by default.



If you set the `batch` property, do not set the `command` or `commandParameter` properties.

## Views

- **FooterView:** None
- **FooterMenuView:**
  - `menuId`: The ID of the menu to display in `FooterMenuView`. The menu is configured in the Footer module's `menus` property.

## View Models

- **FooterViewModel:**
  - `backgroundColor`: A valid HTML color to use for the footer's background. For example, **white** or **#FFFFFF**.



Footer text is black and hyperlinks are dark blue, so use a light color for the background.

- `height`: The footer's height, in pixels. When the height is 0 (zero), the viewer does not have a footer. By default, the height is 0.
- **FooterMenuViewModel:** None

## Example 1: Show Text Hyperlinks in the Footer

This example shows how to configure a menu of text hyperlinks.

First, define the menu. The following snippet shows the configuration for a footer menu that contains two text items.

```
...
{
  "id": "FooterMenu",
  "description": "@language-menu-footer-menu-desc",
  "moduleId": "Footer",
  "items": [
    {
      "text": "@language-menu-powered-by-geocortex",
      "description": "@language-menu-powered-by-geocortex-desc",
      "command": "OpenWebPage",
      "commandParameter": "http://www.geocortex.com/"
    },
    {
      "text": "Report a Problem",
      "command": "RunWorkflowById",
      "commandParameter": "ShowProblemReportForm"
    }
  ]
},
...
```



Next, define the view. To ensure that the items appear as hyperlinks rather than simple text, set the view's `markup` property to `Mapping/infrastructure/menus/MenuHyperlinkView.html`. Set the `region` property to the region where you want the menu to appear, in this case, `RightFooterRegion`. Set the `menuId` property to the menu's ID, `FooterMenu`.

```
...
{
  "id": "FooterMenuView",
  "viewModelId": "FooterMenuViewModel",
  "libraryId": "Mapping.Infrastructure",
  "type": "geocortex.essentialsHtmlViewer.mapping.infrastructure.menus.MenuView",
  "markup": "Mapping/infrastructure/menus/MenuHyperlinkView.html",
  "region": "RightFooterRegion",
  "visible": true,
  "configuration": {
    "menuId": "FooterMenu"
  }
}
...
```

Finally, set the `FooterViewModel`'s `height` property and, optionally, the `backgroundColor` property.

```
...
{
  "id": "FooterViewModel",
  "type": "geocortex.essentialsHtmlViewer.mapping.modules.footer.FooterViewModel",
  "configuration": {
    "backgroundColor": "white",
    "height": 40
  }
}
...
```

## Example 2: Show Image Items in the Footer

This example shows how to configure a menu of image items.

First, define the menu. The following snippet shows the configuration for a footer menu with three items. The first item does not run a command, so it appears as a simple (unclickable) image in the footer. The other two items run commands, so they are hyperlinks.

```
...
{
  "id": "SocialFooterMenu",
  "description": "@language-menu-social-links-menu-desc",
  "moduleId": "Footer",
  "items": [
    {
      "iconUri": "Resources/Images/Icons/logo-24.png",
    },
    {
      "iconUri": "Resources/Images/Icons/twitter-24.png",
      "command": "ShareOn",
      "commandParameter": "twitter"
    },
    {
      "iconUri": "Resources/Images/Icons/Toolbar/contact-24.png",
      "command": "ShareOn",
      "commandParameter": "email"
    }
  ]
}
...
```

Next, define the view. Set the view's `markup` property to `Mapping/infrastructure/menus/MenuView.html`. Set the `region` property to the region where you want the menu to appear, in this case, `LeftFooterRegion`. Set the `menuId` property to the menu's ID, `SocialFooterMenu`.

```
...
{
  "id": "SocialFooterMenuView",
  "viewModelId": "SocialFooterMenuViewModel",
  "libraryId": "Mapping.Infrastructure",
  "type": "geocortex.essentialsHtmlViewer.mapping.infrastructure.menus.MenuView",
  "markup": "Mapping/infrastructure/menus/MenuView.html",
  "region": "LeftFooterRegion",
  "visible": true,
  "configuration": {
    "menuId": "SocialFooterMenu"
  }
}
...
```

Next, add a new `MenuViewModel` whose `id` corresponds to the `viewModelId` which, in the above example, is named `SocialFooterMenuViewModel`.

```
...
{
  "id": "SocialFooterMenuViewModel",
  "type":
"geocortex.essentialsHtmlViewer.mapping.infrastructure.menus.MenuViewModel",
  "configuration": {}
}
...
```

Finally, set the `FooterViewModel`'s `height` property and, optionally, the `backgroundColor` property.

```
...
{
  "id": "FooterViewModel",
  "type": "geocortex.essentialsHtmlViewer.mapping.modules.footer.FooterViewModel",
  "configuration": {
    "backgroundColor": "#EEEEEE",
    "height": 50
  }
}
...
```

### Example 3: Show the Scale Bar and Map Coordinates in the Footer

You can show any module in a viewer's footer by setting the view's `region` property to one of the footer regions.

For example, to show the scale bar in the footer, you could set the `region` property in the Scalebar module's `ScalebarView` to `LeftFooterRegion`. If you also set `CoordinatesView`'s `region` to `LeftFooterRegion` in the Coordinates module, both the scale bar and the coordinates widget will show in the left part of the footer. Showing the scale bar and coordinates widget in the footer reduces the number of widgets on the map.

## 15.25 Geolocate Module

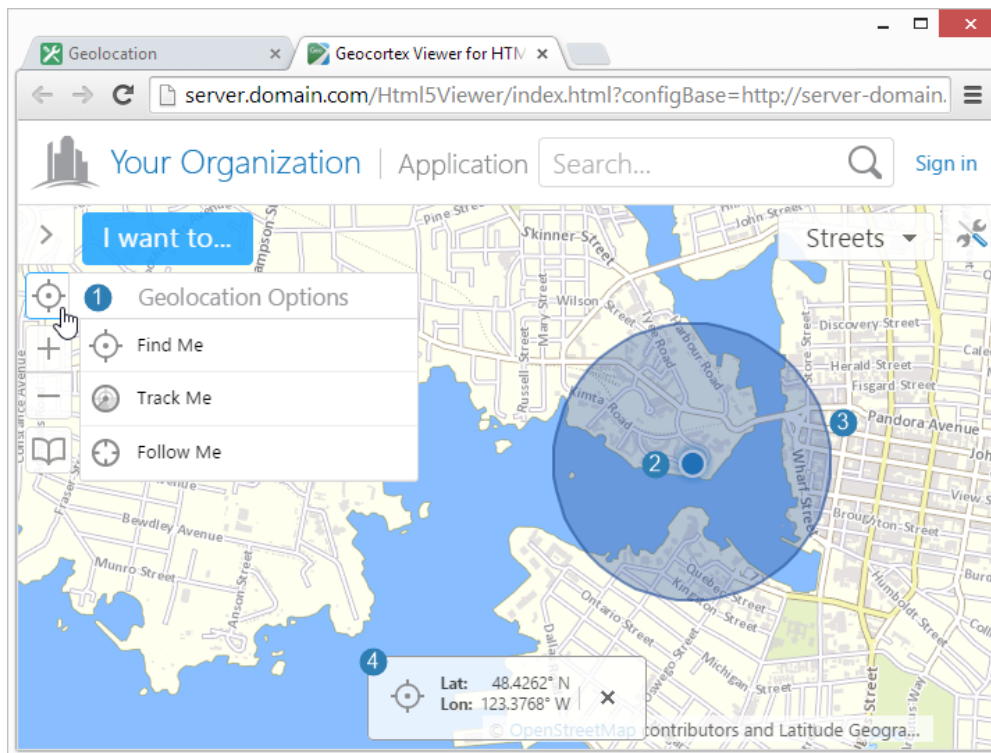


Most of the Geolocate Module's properties can be configured using Manager. For instructions, see [Configure Geolocation on page 95](#).

The Geolocate Module implements geolocation in the Geocortex Viewer for HTML5.

Geolocation locates the user on the map. The HTML5 Viewer supports the following geolocation options:

- **Find Me:** Pans the map to the user's location and marks the location with an indicator.
- **Track Me:** Tracks the user's location with an indicator, without panning the map.
- **Follow Me:** Follows the user's location with an indicator and pans the map as the user's location changes.



Geolocation menu (1), location indicator (2), accuracy circle (3), and geolocation coordinates (4)

## Browser Support

Web browsers return geolocation results in WGS 84. If your map is not in Web Mercator or WGS 84, you must configure an ArcGIS geometry service so the geolocation widget can project from latitude/longitude to the map's coordinates. For instructions on configuring a geometry service, see [Configure Application-Wide Settings on page 70](#).

Each web browser handles geolocation differently. For example, as of Chrome 50.0, your site needs to be secured with HTTPS in order to use the Geolocation API. If you or your users intend to use geolocation, ensure that your preferred web browser uses geolocation and that it is compatible with your site. Here are two common scenarios where geolocation is not available to users:

- **Chrome 50+:** Your site needs to be secured with HTTPS in order to use the Geolocation API.
- **Internet Explorer 8:** Geolocation is not supported in Internet Explorer 8 or older. If you use Internet Explorer Enterprise Mode, you may be emulating a version of Internet Explorer 8.



It is common for devices to ask permission before websites can use geolocation. If geolocation is not working, ensure that the viewer has been given sufficient geolocation privileges. Also ensure that the device's location services are enabled.

In order for an HTML5 viewer to perform geolocation operations, the user's device must have a built-in GPS (Global Positioning System), a Wi-Fi connection, or a wired connection. GPS provides the most accurate geolocation results. Wi-Fi is less accurate than GPS, and geolocation using a wired connection is less accurate than Wi-Fi.

Some devices use a mix of GPS and Wi-Fi to obtain a location. This can result in unpredictable readings. Accuracy is improved by setting the device to GPS-only mode, however, this drains the battery more quickly in some devices.

## Configuration Properties

### Module

None

### Views

- **GeolocateView:** None
- **GeolocateStatusView:** None

### View Models

- **GeolocateViewModel:**
  - **geolocateEnabled:** To display the option for single-reading geolocation (**Find Me**), set to `true`; otherwise, set to `false`. The default is `true` for the Tablet and Handheld interfaces, and `false` for the Desktop interface. When used, the map pans to the user's current location, as indicated by a marker.
  - **trackingEnabled:** To display the option for tracking the user's location (**Track Me**), set to `true`; otherwise, set to `false`. The default is `true` for the Tablet and Handheld interfaces, and `false` for the Desktop interface. When used, the user's location is tracked with a marker, without panning the map.
  - **followingEnabled:** To display the option for following the user's location (**Follow Me**), set to `true`;

otherwise, set to `false`. The default is `true` for the Tablet and Handheld interfaces, and `false` for the Desktop interface. When used, the user's location is followed with a marker, and pans the map as the user's location changes.

- **enableHighAccuracy**: To enable have the GPS provide the most accurate position possible, set to `true`; otherwise, set to `false`. The default is `true`. Setting this property to `true` may result in slower response times or increased power consumption.
- **singleGeolocationProfiles**: An object whose properties define a number of named single-reading geolocation profiles. By default, these profiles are: `default`, `coarse` and `fine`. A profile has the following properties:
  - **accuracyThreshold**: A number representing the accuracy radius, in meters, that satisfies a single-reading geolocation.
  - **timeLimit**: A number representing the duration, in milliseconds, to refine the accuracy of a single-reading geolocation.
- **geolocateAccuracyCircleEnabled**: To enable the geolocation accuracy circle, set to `true`; otherwise, set to `false`. The default is `true`. When enabled, the marker for the user's position includes a surrounding circle that indicates the margin of error of the user's position; the user's actual position should lie somewhere within this circle.
- **adjustExtentZoomOnGeolocate**: To enable zooming to the user's location upon single-reading geolocation and following, set to `true`; otherwise set to `false`. The default is `true`.
- **geolocateExtentZoomLevel**: The scale to which to zoom when `adjustExtentZoomOnGeolocate` is set to `true`. The default is `50000` (to 1).
- **geolocationIndicator**: The URL to the image that represents the geolocation indicator. If this property is omitted, the default is an image of a blue dot located at `Resources/Images/Icons/geolocate-position-32.png`.
- **GeolocateStatusViewModel**:
  - **showGeolocateCoordinates**: When this property is `true`, **Find Me** operations show the coordinates of the user's location, in addition to showing the geolocation indicator. If you do not want to show the coordinates for Find Me operations, set `showGeolocateCoordinates` to `false`.
  - **showTrackingCoordinates**: When this property is `true`, **Track Me** operations show the coordinates of the user's location, in addition to showing the geolocation indicator. If you do not want it show the coordinates for Track Me operations, set `showTrackingCoordinates` to `false`.
  - **showFollowingCoordinates**: When this property is `true`, **Follow Me** operations show the coordinates of the user's location, in addition to showing the geolocation indicator. If you do not want to show the coordinates for Follow Me operations, set `showFollowingCoordinates` to `false`.
  - **coordinateFormat**: The units to use for displaying the coordinates. The options are:
    - **Degrees/Decimal Minutes**: Set `coordinateFormat` to `ddm`.
    - **Degrees/Minutes/Seconds**: Set `coordinateFormat` to `dms`.
    - **Latitude/Longitude**: Set `coordinateFormat` to `dd`.

- **X/Y:** Set `coordinateFormat` to `xy`. By default, X/Y coordinates are shown in the map's spatial reference. To change the spatial reference that is used for X/Y coordinates, configure the `coordinateWkid` property.
- **coordinateWkid:** The well-known ID of the coordinate system to use for X/Y coordinates.
- **coordinateFractionalDigits:** The number of decimal digits to show in the coordinates.



This precision setting cannot exceed the maximum representable IEEE 754 floating point. This is specified in the IEEE 754 standard. See [double-precision floating-point format](#) and [significant figures](#) for more information.

- **geolocateIcon:** The icon to show beside the coordinates for Find Me operations.
- **busyIcon:** The icon to show beside the coordinates when the geolocation widget is tracking or following the user.

See also...

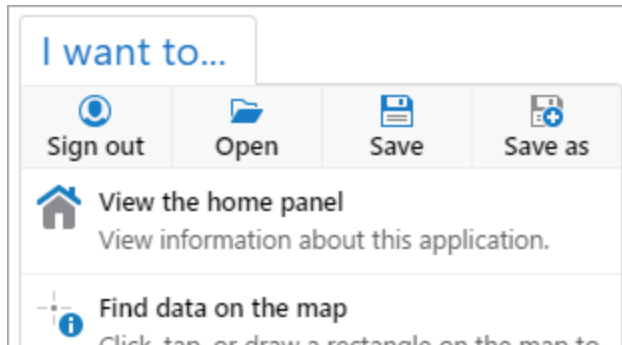
[Configure Geolocation on page 95](#)

[Configure Application-Wide Settings on page 70](#)

[Navigation Module on page 285](#)

## 15.26 GlobalMenu Module

The GlobalMenu Module implements a submenu at the top of the I Want To menu.



### Global menu in the I Want To menu

By default, the following menu items are included:

- **Sign in / Sign out:** Sign into or sign out of the viewer.
- **Open:** Open the Projects panel.

The Projects panel lists the projects that this user is allowed to load. The user clicks a project in the list to load the project. This restores the viewer session that is saved in the project.

- **Save:** Save a project.

If a project is currently loaded and the user has the necessary privileges, update the project with the current

viewer session. Otherwise, create a new project using the name and description entered by the user.

- **Save as:** Create a new project using the name and description entered by the user.

For information about the project options (Open, Save, Save as), see [Project Module on page 304](#). The Sign in and Sign out options perform the same functions as the Sign in and Sign out links in the viewer's banner. For more information, see [User Module on page 389](#).

If you do not want the Global menu to show in the I Want To menu, you can turn it off using the IWantToMenu Module's `showGlobalMenu` property. For example, you might want to hide the Global menu if the viewer does not allow users to sign in. See [IWantToMenu Module on page 215](#) for information.

## Configuration Properties

### Module

- **menus:** An array of menus to display in the I Want To menu. By default there is one menu, the Global menu:
  - **id:** The menu's unique ID. The default menu's ID is `GlobalMenuConfig`.
  - **defaultIconUri:** The icon to use when a menu item's `iconUri` property is not configured.
  - **items:** An array of menu items. Menu items have the following properties:
    - **iconUri:** The image to display on the menu item. The recommended image size is 24px by 24px.
    - **text:** The text to display on the menu item. You can use a text key or the literal text.



If your viewer is going to be available in more than one language, enter the text key that the description is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **description:** The tool tip to display when the user positions the pointer over the menu item.
- **command:** The command to execute when the user clicks the menu item. For more information about commands and events, refer to the Geocortex SDK for HTML5 API Reference. Instructions for accessing the API Reference are [here](#).

- **hideOnDisable:** If this property is set to `true` and the menu item's command cannot run, the menu item does not show in the menu.

If `hideOnDisable` is `false` and the menu item's command cannot run, the menu item shows in the menu, but it is grayed out.

For example, the `SignIn` command cannot run when the user is already signed in, so the **Sign in** menu item is either hidden (if `hideOnDisable` is `true`) or grayed out (if `hideOnDisable` is `false`).

### Views

- **GlobalMenuView:**
  - **menuId:** The ID of the menu to display in this view. The menu is defined in the module's `menus` property. The default is `GlobalMenuConfig`.



## View Models

- `GlobalMenuViewModel: None`

## See also...

[IWantToMenu Module on page 215](#)

[Project Module on page 304](#)

[User Module on page 389](#)

## 15.27 HeatMaps Module

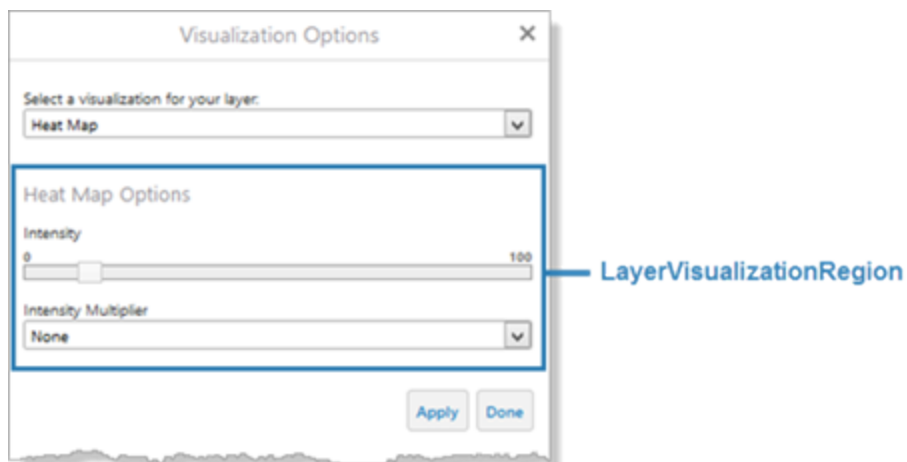
Heat mapping is a data visualization that represents the density of features on the map. The HeatMaps Module works with a number of other modules to implement heat maps. For information about configuring heat maps for a layer, refer to the *Geocortex Essentials Administrator Guide*.

Specifically, the HeatMaps Module implements the `visualizationProvider` for heat maps and the commands and events that work with heat maps. The HeatMaps Module uses the data from the feature layer to calculate the heat map and renders the heat map on the feature layer.

The HeatMaps Module provides two commands and two events, which you can use in hyperlinks and workflows. The `AddHeatMap` command renders a heat map for the specified feature layer. `HeatMapAddedEvent` is raised when a heat map is rendered on the map. The `RemoveHeatMap` command removes the heat map from the specified feature layer. `HeatMapRemovedEvent` is raised when a heat map is removed. At most one visualization can be active at a time. For more information about commands and events, refer to the Geocortex SDK for HTML5 API Reference. Instructions for accessing the API Reference are [here](#).

The HeatMaps Module works with the Menu Module and Visualization Module. The [Menu Module](#) has a `LayerActions` menu item, **Turn on/off layer visualizations**, that opens the Visualization Options panel, where the user can change visualizations and their settings. The Visualization Options panel is implemented by the [Visualization Module](#).

The HeatMaps Module has one view, `HeatMapView`. `HeatMapView` presents the heat map settings that end users can configure. By default, `HeatMapView` is hosted in the `LayerVisualizationRegion` container region, which is referenced by the Visualization Module's `VisualizationViewModel`.



The HeatMaps Module's `intensity` and `gradientOptions` properties set the defaults to use when a layer's heat map settings are not configured in the site. Every feature layer has a Feature Heat Maps tab in Manager where you can configure the heat map settings for that layer. If a user views the heat maps for two layers at the same time, having different gradients for the different layers makes it possible to distinguish one layer's hot areas from the other layer's hot areas. For more information, refer to the *Geocortex Essentials Administrator Guide*.



Enterprise mode emulates Internet Explorer 8, which does not support heat maps.

## Configuration Properties

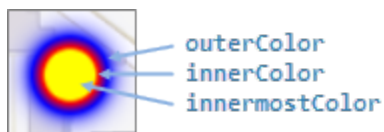
### Module

- `intensity`**: The default area for the hot areas in a heat map to cover, in pixels. The greater the intensity, the larger the area of coverage. Specify a number from 1 to 100. The default is **30** pixels.  
 The value that you configure here is used for layers whose Feature Heat Maps settings are not configured. For information on configuring the settings for a particular layer, refer to the *Geocortex Essentials Administrator Guide*.



Use a lower intensity for feature layers with a high concentration of data and higher intensity for feature layers with sparse data.

- `gradientOptions`**: The default colors that represent the hot areas on the map, and the opacity of each color. The viewer creates gradients using the colors that you configure.  
 The values that you configure here are used for layers whose Feature Heat Maps settings are not configured. For information on configuring the settings for a particular layer, refer to the *Geocortex Essentials Administrator Guide*.
  - `outermostColor`**: The outermost color in the gradient is completely transparent by default, to ensure that the outer color changes gradually. We recommend using the default value for `outermostColor`.
  - `outerColor`, `innerColor`, `innermostColor`**: For each color that you want to use to represent hot areas on the map, specify the color's ARGB hex string. For example, `#FF9D9D9D` is a completely opaque (FF) shade of gray (9D9D9D).



Default gradient—transparent, blue, red, yellow

### Views

- `HeatMapView`**: None

### View Models

- `HeatMapViewModel`**: None

See Also...

[Menu Module on page 282](#)

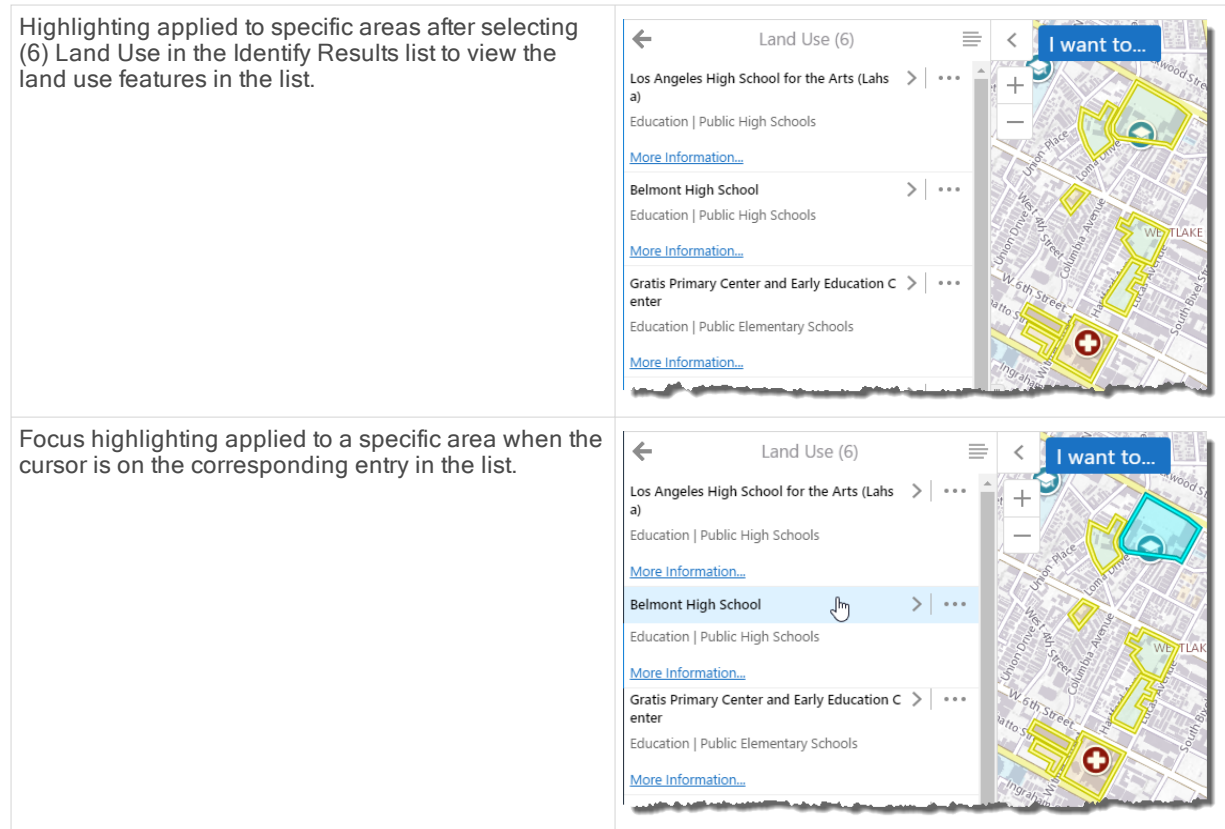
[Visualization Module on page 390](#)

## 15.28 Highlight Module

The Highlight Module configures the fill and border colors used for highlighting and emphasizing features on the map. The viewer initially has a default highlight layer that is used to highlight the feature when the user selects the feature in a Results List. (See [Results Module on page 328](#) for examples of how highlighting is applied to items in the Results List and Results Table.) Highlighting defined for feature layers in Essentials Manager overrides the feature layer highlight settings specified in the Viewer.

For example:

<p>Using Identify to select an area.</p>	
<p>The Identify Results list from the Identify action.</p>	
<p>Highlighting applied to the hospital icon after selecting (1) Hospitals in the Identify Results list to view the hospital feature.</p>	
<p>Focus highlighting applied to the hospital icon when the cursor is on the hospital entry in the panel.</p>	



## Configuration Properties

### Module

- **fillColor:** The fill color specified for highlighting, represented as an RGBA color. The default color is pale yellow defined as "RGBA(236, 236, 58, 0.1)". The first set of digits (236) is for the red color value, the second set (236) is for the green color value, the third set (58) is for the blue color value, and the last value (0.1) is for opacity, where 0.0 is fully transparent, and 1.0 is fully opaque.
- **outerBorderColor:** The outer border color specified for highlighting. The default color is dark yellow, defined as "RGBA(200, 200, 0, 1)".
- **borderColor:** The border color specified for highlighting. The default color is yellow, defined as "RGBA(255, 255, 151, 1)".
- **focusedFillColor:** The fill color specified when the highlighted feature has focus. The default color is pale cyan, defined as "RGBA(0, 255, 255, 0.2)".
- **focusedBorderColor:** The border color specified when the highlighted feature has focus. The default color is cyan, defined as "RGBA(0, 255, 255, 1)".
- **focusedOuterBorderColor:** The outer border color specified when the highlighted feature has focus. The default color is dark cyan, defined as "RGBA(87, 170, 255, 1)".
- **outerBorderWidth:** The outer border width specified for highlighting. The default value is 5 pixels.
- **borderWidth:** The border width specified for highlighting. The default value is 2 pixels.

- **highlightLineOpacity:** The opacity setting for a highlighted line feature. The default is 0.5.
- **maxHighlightableGeometryVertices:** The maximum number of vertices that can be highlighted before issuing a warning message. If the number of vertices exceeds this value, the message indicates that geometries in excess of "n" vertices cannot be highlighted. The user can select to not show the message again. The default for the maximum number of vertices is 5000.
- **geometryGeneralization:** Provides options to simplify a large geometry containing several thousand vertices to facilitate the highlighting function.
  - **GeometryGeneralizationEnabled:** The setting that turns geometry generalizing on or off. The default value is true.
  - **thresholdVertices:** Indicates the threshold at which geometry generalizing is applied. The default threshold value is 5000 vertices.
  - **maxDeviationInMeters:** Defines the maximum deviation to apply when simplifying the original geometry. The default value is 250 meters.

#### Views

None.

#### View Models

None.

## 15.29 Identify Module

The Identify Module implements identify operations. It provides a set of configurable tools, each of which runs an `Identify` command on a particular type of geometry. The tools return a collection of features that intersect the geometry that the user draws on the map. The feature set collection is then used by some other module, which renders the feature set collection in its view.

For example, the [MapTips Module](#) uses the `Identify` command to identify a feature located at a particular point. The MapTips view then renders the feature returned by the `Identify` command.



WMS layers that are not associated with a WFS only support point identify operations, whereas WMS layers associated with a WFS support all types of identify operations.

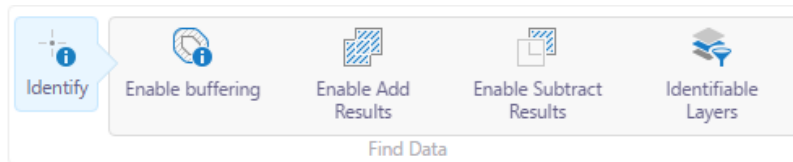
As of HTML5 Viewer 2.5, the user can choose which layers are affected by the Identify tools by clicking the **Identifiable Layers** button. Users can also enable [buffering](#) to identify nearby features.



The **Identifiable Layers** button is configured in context-sensitive toolbars within the [TabbedToolbar](#) and [CompactToolbar](#) modules. The easiest way to configure this feature is to edit your viewer in Essentials Manager, and navigate to the **Toolbar** section. To enable this feature for the stand-alone **Identify** tool, ensure both **Identify** and **IdentifyToolControlRegion** are included in your configured toolbar. To enable this feature for the **Identify** multitool, ensure both **Find Data** and **FindDataControlRegion** are included in your configured toolbar. In the case of the Compact Toolbar, do not include the regions.

The Identify multitool also provides options to add to existing identify results or to subtract from existing identify results. (These options are not currently available for handheld devices.) To add to existing results, click **Identify** to open the menu, click **Enable Add Results**, then click another item on the map to add it to the results list. Similarly, to subtract from existing results, click **Identify** to open the menu, click **Enable Subtract Results**, then click an item previously identified to remove it from the results list.

By default, **Identify** is turned off after each use. To continue to add to or subtract from the identify results, click **Identify** and click or tap another location on the map. When the user closes the Results List or Table, the features that were listed are discarded.



### The Identify multitool showing the Enable Add Results and Enable Subtract Results options

Note that if **Enable Add Results** is active and the user clicks **Enable Subtract Results**, the add results option is disabled. Similarly, if **Enable Subtract Results** is active and the user clicks **Enable Add Results**, the subtract results option is disabled.

## Configuration Properties

### Module

- **resultsDisplayName:** The title of the Results List that displays after an identify operation. The default is `@language-identify-results-header`.
- **topMostLayerOnly:** When `true`, the feature set collection that is returned by the identify operation includes features from the topmost layer only. The default is `false`.
- **visibleLayersOnly:** To restrict the identify operation to perform only on visible layers, set to `true`; otherwise, set to `false`. The default is `true`.
- **layersInVisibleScaleRangeOnly:** To restrict the identify operation to perform only on layers within the visible scale range, set to `true`; otherwise, set to `false`. The default is `true`.
- **pixelTolerance:** A positive integer that defines the maximum number of pixels away from a feature the user can click with a point-based Identify tool in order to identify the feature. The default pixel tolerance is 5 pixels.
- **polygonPixelTolerance:** A positive integer that defines the maximum number of pixels away from a feature the user can draw a shape with a polygon-based Identify tool in order to identify the feature. The default polygon pixel tolerance is 0 pixels, in other words, the user must draw a shape that directly overlaps at least part of the feature.
- **returnGeometry:** When this property is `true`, the identify operation returns a geometry. This is useful for feature actions that require the geometry, like `ZoomToFeature` or `PanToFeature`. The default is `true`.
- **restrictRasterIdentifyToPoint:** When this property is `true`, only point-identify operations return results from image services and raster layers. If the user performs some other type of identify, such as a rectangle identify, no results are returned from image services and raster layers. By default, `restrictRasterIdentifyToPoint` is `true`.

If you want all identify operations to return results from image services and raster layers, set this property to `false`. This is useful when the user wants to get a sense of what's in an area—the user can perform a single identify operation that returns results from multiple feature layers, images services, and raster layers. Note that line-based and polygon-based identify operations return the results for a single point in raster layers and image services. Line-based identify operations return the values for the line's midpoint. Polygon-based identify operations return the values for the polygon's centroid.

- **identifyProviders:** An array of identify providers. The default providers are:
  - **GraphicsLayerIdentifyProvider:** This provider enables identify operations on graphics layers, such as the markup layer, collaboration layers, and ArcGIS Stream Layers.
  - **MapIdentifyTaskIdentifyProvider:** This provider enables identify operations on feature layers.
  - **RasterIdentifyTaskIdentifyProvider:** This provider enables identify operations on image services and raster layers.
- **tools:** An array of tools that perform identify operations. The factory configuration has tools to identify by point, by rectangle, by polyline, by polygon, and by freehand polygon.

The [Tools Module](#) implements the ability to define an array of tools in other modules. Each tool in the array has the following properties:


- **name:** The name of the tool.





If your viewer is going to be available in more than one language, enter the text key that the tool name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **command:** The command that the tool runs.  
For a list of commands, see the [Geocortex SDK for HTML5 API Reference](#).
- **drawMode:** The type of geometry the user draws, upon which the tool operates.
- **isSticky:** When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.  
If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
- **iconUri:** The image that displays beside the tool.
- **statusText:** The status message to display when the tool's input method is via mouse, often containing instructions for the user. You can use a text key or the literal text.
- **keyboardStatusText:** The status message to display when the tool's input method is via keyboard, often containing keyboard shortcut hints for the user. You can use a text key or the literal text.
- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **SelectLayersForIdentifyActivatedBehavior:** A behavior that runs an array of commands when the user clicks the **Identifiable Layers** button. By default, this includes the command: `OpenDataFrame`.
  - **SelectLayersForIdentifyDeactivatedBehavior:** A behavior that runs an array of commands

when the **Identifiable Layers** panel is dismissed by a toggle button. By default, this includes the command: `CloseDataFrame`.

 You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.

 You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.

 Adding a new behavior is only recommended for experienced developers.

## Views

- `IdentifyOptionsView`: None
- `IdentifyLayerSelectorView`: None

## View Models

- `IdentifyOptionsViewModel`: None
- `IdentifyLayerSelectorViewModel`: None

## See Also...


[MapTips Module on page 253](#)

[Tools Module on page 375](#)

[About User Interface Text on page 64](#)

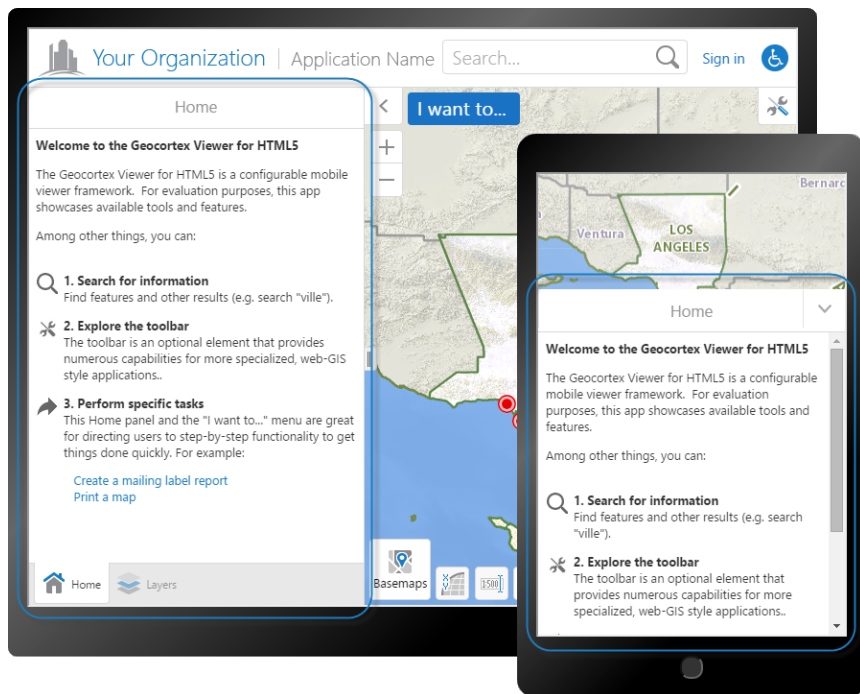
[Geocortex SDK for HTML5 API Reference on page 422](#)

## 15.30 Info Module

 This module can be configured using Manager. For instructions, see [Configure the Home Panel on page 99](#).

The Info Module implements the Home Panel, which acts as an entry point or home base for end users. The contents of the Home Panel can include text, images, and UI elements like buttons and links to launch web pages, invoke commands, and run workflows. The contents are defined in HTML.





### Home Panel in the Desktop interface (rear), and in the Handheld interface

If you want the user to see the Home Panel when the viewer launches, you must configure the following properties:

- In the [Info Module](#)'s `InfoViewModel`, set the `included` property to `true`.
- In the [Shells Module](#), set the `homePanelVisible` property to `true`.
- For the Desktop and Tablet interfaces only, in the [Shells Module](#)'s `ShellViewModel`, set the `dataFrameOpenByDefault` property to `true`.

In the Desktop and Tablet interfaces, the Home Panel is hosted in the Data Frame. Setting `dataFrameOpenByDefault` to `true` ensures that the Data Frame is open when the viewer launches.

The Info Module and Home Panel were introduced in version 1.3 of the HTML5 Viewer.

## Configuration Properties

### Module

None

### Views

- **InfoView:** None

## View Models

- **InfoViewModel:**

- **content:** A string containing the HTML markup that defines the contents to display in the Home Panel.



Essentials HTML-encodes the Home Panel `content` string when you edit and save the viewer configuration in Essentials.



You can use a valid text key for the content as long as there are no spaces or HTML markup surrounding the text key. For more information on using text keys, see [About User Interface Text on page 64](#).

- **included:** When this property is set to `true`, the Home Panel is available in the viewer. When `included` is `false`, the viewer does not have a Home Panel. The default is `true`.
- **title:** A string that defines the title that appears at the top of the Home Panel.



If your viewer is going to be available in more than one language, enter the text key that the Home Panel title is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

The default title is `@language-common-welcome`.

### See Also...

[Shells Module on page 350](#)

[About User Interface Text on page 64](#)

## 15.31 InsightIntegration Module

The `InsightIntegration` Module implements integration between Geocortex Analytics and the HTML5 viewer.

The `InsightIntegration` Module collects usage and other data about the viewer and sends it to the Client Relay Collector in Analytics. The `InsightIntegration` module creates a file in local storage and adds events that happen in the viewer to this file. Once the file reaches 100 KB, or the configured amount of time has lapsed, the module creates a zip file and sends it to the Client Relay Collector.

The `InsightIntegration` Module then collects data about:

- Viewer users
- Map services, layers and areas viewed
- Tools used
- Searches performed
- Datalinks resolved
- Log messages

- Print events
- Report-generation events
- Workflow requests
- Unhandled errors

This data is collected and used in informative reports within Analytics.



The domains of Analytics and the HTML5 Viewer are likely different. If so, to ensure the HTML5 Viewer can send data to Geocortex Analytics, you must add a proxy entry for Geocortex Analytics in the `proxy.config` file, which is located in the root folder where HTML5 Viewer is installed. For example, `<serverUrl url="http://MyDomain.com/Geocortex/Analytics" matchAll="true"></serverUrl>`.

## Configuration Properties

### Module

- **enabled:** If set to `true`, the module is enabled.
- **dataRelayUri:** Sets the URL to the Client Relay in Analytics. For example, `"http://localhost/Geocortex/Analytics/ClientRelay"`
- **dataRelayIntervalInSeconds:** Sets the time interval in seconds that the module waits before sending accumulated data to the Client Relay Collector. The default is **30** seconds.

### Views

- **ExternalComponentView:** None

### View Models

- **ExternalComponentViewModel:**
  - **containerRegionName:** The name of the container region in which to host Analytics integration. The default is `ExternalComponentRegion`.
  - **containerRegionType:** The type of the container region in which to host Analytics integration. The default is `geocortex.framework.ui.DivStackRegionAdapter`.
  - **headerIsVisible:** To display the container header, set to `true`; otherwise, set to `false`. The default is `true`.
  - **showXButton:** To display a button to close the container, set to `true`; otherwise, set to `false`. The default is `true`.
  - **showMaximizeButton:** To display a maximize button for the container, set to `true`; otherwise, set to `false`. The default is `true`.
  - **resizeY:** To allow the container to be vertically resized, set to `true`; otherwise, set to `false`. The default is `true`.

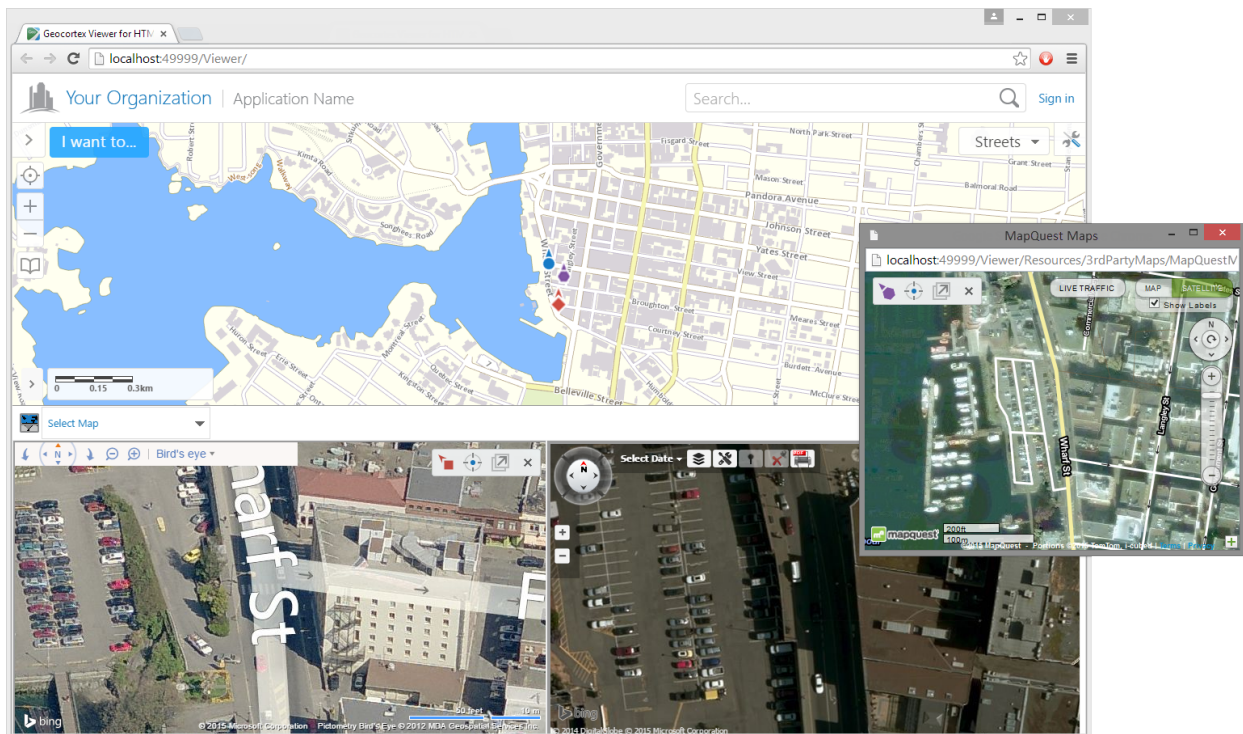
- **selectorIconUri**: (optional) The URI for the icon to display at the left end of the header. The icon is for display only—it is not clickable.
- **selectorText**: The text to label the drop-down list of external components.
- **defaultComponents**: An array of IDs of external components to open when ExternalComponentView is activated. The default is [ ].
- **ExternalComponents**: An array of applications that the viewer integrates with. The default is [ ].

See Also...

[Configure Analytics Integration on page 124](#)

## 15.32 Integration Module

The Integration Module implements the ability to open other applications from within an HTML5 viewer. For example, integrating Bing Maps with an HTML5 viewer enables users to open Bing Maps alongside the viewer. In the screen capture below, three mapping applications are open. Two of the applications are docked to the viewer and one is undocked.



Mapping applications integrated with an HTML5 viewer



You are responsible for ensuring that you have the required licenses for the applications that you integrate.



The Handheld user interface does not support integration.



Internet Explorer's Enterprise mode does not support integration—Enterprise mode emulates Internet Explorer 8, which does not support integration.

To integrate an application with an HTML5 viewer, you configure an **external component** for the application in the viewer. The external component's configuration can either specify the web application to integrate, or it can specify a web page that points to the web application to integrate.

The simplest configuration points directly to a public web application (or website). In this case, the application displays in an integration pane without any additional controls beyond what the application itself offers. To configure this, all you have to do is configure the external component to point to the web application. End users will be able to open the application from the viewer and use the application's controls.

You may want to provide additional controls for interacting with an integrated application. For example, when you integrate a mapping application, you may want users to be able to center the integrated map on the viewer's map. In this case, you host a web page that points to the application that you want to integrate and you build the additional controls into the web page. You must also provide any supporting files that the web page references, such as CSS and JavaScript files. When you configure the external component in the viewer, you specify the web page that points to the application, instead of the application itself.



If the application is licensed, you may need to host a web page that provides the licensing information, instead of pointing directly to the application.

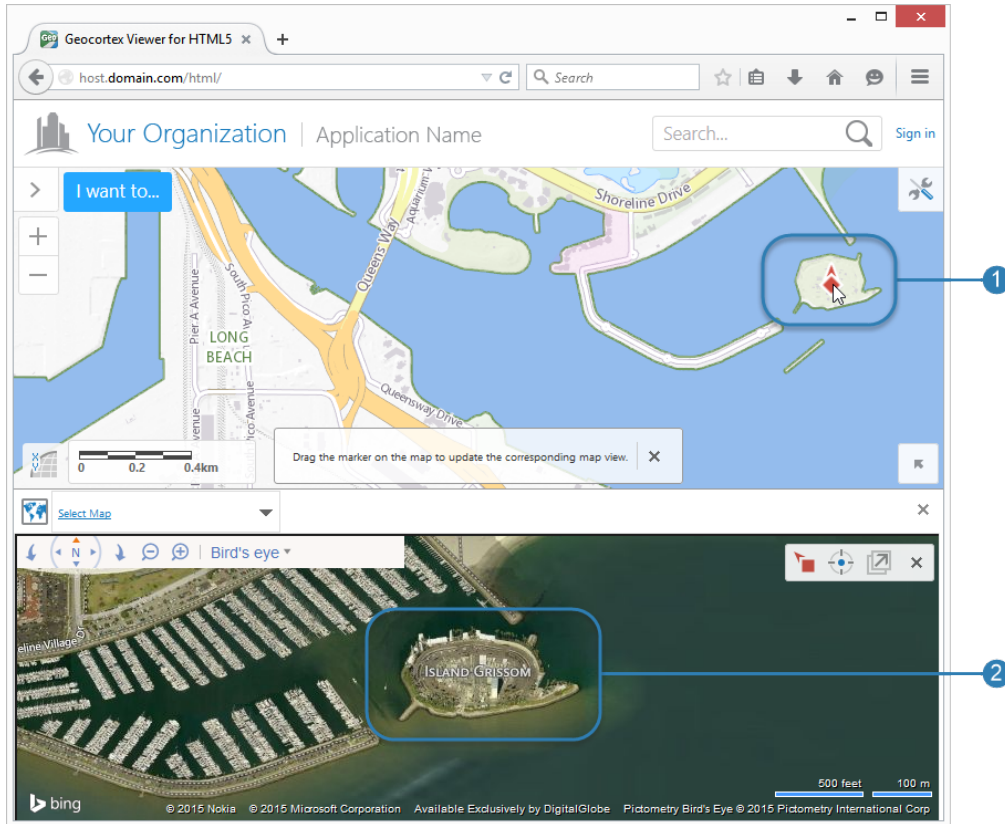
The HTML5 Viewer ships with integration samples for Bing Maps and Pictometry that you can use. See [Example 1: Integrate with Bing Maps on page 207](#) and [Example 2: Integrate with Pictometry on page 211](#) for instructions. As well, the Geocortex Support Center's Code Gallery includes a [sample that integrates Google Street View](#).

## Integration with Mapping Applications




Effective with version 2.9 of the HTML5 Viewer, Internet Explorer version 11 and later is required for third-party mapping applications.

When an integrated map is open, the viewer's map displays a marker that indicates the center of the integrated map. Dragging the marker pans the integrated map so that the map stays centered at the center of the marker's body.






**Dragging the marker (1) pans the integrated map (2) so it stays centered on the marker**

You can pan and zoom the viewer's map and integrated maps independently of each other by using the standard map controls for panning and zooming. If you pan or zoom an integrated map, the marker stays synchronized with the map.

You can configure a different marker for each integrated map . Using different markers enables users to distinguish one map's marker from another's when there is more than one integrated map open.

The HTML5 Viewer ships with integration samples for Bing Maps and Pictometry. The sample web pages contain tools for managing the integrated map and its pane independently of any other integrated applications that are open. The tools are:

-  **Viewpoint Indicator:** Shows the marker that is used for this integrated map. The Viewpoint Indicator is for information only—it is not clickable.  
 The Viewpoint Indicator is configured in the CSS file that is provided with the sample. If you change a map's marker in the Integration Module, you must also update the Viewpoint Indicator in the CSS.
-  **Center:** Moves the integrated map's marker to the center of the viewer's map and rescales the integrated map to match the viewer's map as closely as possible.
-  **Dock/Undock:** (Desktop interface only) Allows the user to dock and undock the integrated map from the viewer. Depending on the browser settings, the map may undock to a new tab or to a new window. The Tablet shell does not support undocking integration panes.



Internet Explorer 9 does not support undocking integration panes.

- **Close:** Allows the user to close the current integration pane without closing any other panes that are open.

## Open an Application from an HTML5 Viewer

The Integration Module has one view—`ExternalComponentView`. By default, `ExternalComponentView` is not visible, so you must provide a way for users to activate the view. The HTML5 Viewer provides a **Linked Maps** button that you can add to the toolbar. Alternatively, you could add an I Want To menu item or a hyperlink that runs one of the activation commands. You can also use these commands in workflows.

The commands that activate `ExternalComponentView` are `ShowExternalComponentView`, `DisplayDockedExternalComponentById`, and `DisplayUndockedExternalComponentById`.

By default, the Linked Maps button uses the `ShowExternalComponentView` command.

`ShowExternalComponentView` opens each default component (`defaultComponents`) that is configured. Each component opens in its own pane. If there are no `defaultComponents` configured, `ShowExternalComponentView` opens the first component that is configured in `externalComponents`.

The `DisplayDockedExternalComponentById` and `DisplayUndockedExternalComponentById` commands open the component that you specify as the command's parameter.

## Configuration Properties

### Module

- **allowedOrigins:** An array of the origins of the `externalComponents` that are on a different domain than the viewer, and that use HTML5 Viewer commands.

In addition to configuring the external component's origin, you must configure the viewer's origin in the `ThirdPartyMap.js` file that ships with the Viewer. Configuring the origins enables two-way communications between the viewer and the integration pane. For more information, see [Configure Cross-Domain Access for an Integrated Application on page 205](#).

This property is an array. To configure two or more items in an array, separate the items with commas. For example:

```
"allowedOrigins": ["http://mymaps.mydomain.com", "https://maps.anotherdomain.com"]
```

If the application does not use any HTML5 Viewer commands, or it is hosted on the same domain as the viewer, then you do not need to configure any `allowedOrigins`.

### Views

- **ExternalComponentView:** None

## View Models

- **ExternalComponentViewModel:**

- **containerRegionName:** The name of the region that hosts this view container. The default is `ExternalComponentRegion`.
- **containerRegionType:** The type of region that hosts this view container. The default is `geocortex.framework.ui.DivStackRegionAdapter`.
- **headerIsVisible:** Specifies whether to show the header at the top of the docking area. The header contains an optional icon (`selectorIconUri`), the drop-down list of `externalComponents` labeled with `selectorText`, and an optional Close button (`showXButton`).  
If you are going to configure only one external component, you can set `headerIsVisible` to `false`. By default, `headerIsVisible` is `true`.
- **showXButton:** When set to `true`, the header displays a button with an X on it that the user can click to close the integration panes. By default, `showXButton` is `true`.
- **showMaximizeButton:** To display a maximize button for the container, set to `true`; otherwise, set to `false`. Maximizing the container makes the integration pane fill the entire screen. The default is `true`.
- **resizeY:** To allow the container to be vertically resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **selectorIconUri:** (optional) The URI for the icon to display at the left end of the header. The icon is for display only—it is not clickable.
- **selectorText:** The text to label the drop-down list of external components.
- **statusText:** The text to display on the map when integration is activated. The default text gives instructions for navigating integrated maps. If you do not want to display any text, set `statusText` to an empty string: `"statusText" = ""`.
- **defaultComponents:** An array of IDs of external components to open when `ExternalComponentView` is activated.

This property is an array. To configure two or more items in an array, separate the items with commas. For example:

```
"defaultComponents": ["bingMaps", "pictometry"]
```

- **externalComponents:** An array of applications that the viewer integrates with. When `headerIsVisible` is `true`, the integration pane's header displays a drop-down list of `externalComponents`. Each application in the array has the following properties:
  - **id:** The unique ID that identifies this external component. The ID is a string.
  - **displayName:** The name to display in the drop-down list of external components. The user selects a component from the drop-down list to open the application in a new pane. The display name can contain spaces and special characters.
  - **uri:** The URI of the application or the hosted web page.  
If the web page is hosted on a different domain than the viewer and it uses one or more HTML5 Viewer commands, then you must configure cross-domain access. See [Configure Cross-Domain Access for an Integrated Application on page 205](#) for instructions.



- **viewpointIndicatorUri:** The URI for the marker that appears on the viewer's map to mark the location of this external component's map.

Markers are provided in the `Resources` folder of the viewer's virtual directory:

`Resources/Images/Icons/location-direction-[color]-32.png`

This property is used with mapping applications only.

This property is an array. To configure two or more items in an array, separate the items with commas. For example:

```
"externalComponents": [
  {
    "id": "bingMaps",
    "displayName": "Bing Maps",
    "uri": "Resources/3rdPartyMaps/BingMaps.html",
    "viewpointIndicatorUri": "Resources/Images/Icons/location-direction-
red-32.png"
  },
  {
    "id": "pictometry",
    "displayName": "Pictometry",
    "uri": "Resources/3rdPartyMaps/Pictometry.aspx",
    "viewpointIndicatorUri": "Resources/Images/Icons/location-direction-
purple-32.png"
  }
]
```

## Configure Cross-Domain Access for an Integrated Application

Configuring cross-domain access enables the viewer and integration pane to communicate with each other. You must configure cross-domain access if all of the following conditions are met:

- you are hosting a web page that points to the application, instead of pointing directly to the application; and
- the web page is deployed to a different domain than the viewer; and
- the web page uses one or more HTML5 Viewer commands.

You do not need to configure cross-domain access if the viewer and web page are deployed to the same domain, or if you are pointing the viewer configuration directly to the application, or if you are pointing the viewer to a web page that does not use any HTML5 Viewer commands.

### ► To configure cross-domain access for an integrated application:

#### Step 1: Configure the web page's origin in the Integration Module

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js` or `Tablet.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. Add the web page's origin to the Integration Module's `allowedOrigins` property.

For example, if the web page is deployed to `http://mymaps.mydomain.com`:

```
{
  "moduleName": "Integration",
  "moduleType":
"geocortex.essentialsHtmlViewer.mapping.modules.integration.IntegrationModule",
  "configuration": {
    "allowedOrigins": ["http://mymaps.mydomain.com"]
  }
  ...
}
```



The origin that you configure in the `allowedOrigins` property must be identical to the URL that you configure in the externalComponent's `uri` property. If the web page, `webpage.html`, is deployed to `http://mymaps.mydomain.com`, then the `uri` property for the external component looks like this:

```
"externalComponents": [
  {
    ...
    "uri": "http://mymaps.mydomain.com/maps/webpage.html",
    ...
  }
]
```

4. Save the file.
5. Repeat these steps for the other configuration file.

## Step 2: Configure the viewer's origin in `ThirdPartyMap.js`

1. Run an HTML or text editor as an administrator.
2. Open `ThirdPartyMap.js` in the editor.

In the default deployment, the file is here:

```
C:\inetpub\wwwroot\Html5Viewer\Resources\3rdPartyMaps\ThirdPartyMap.js
```

3. Find the following line:

```
this.allowedOrigins = [];
```

4. Type or paste the viewer's origin in the array, within quotation marks:

```
this.allowedOrigins = ["http://myviewers.mydomain.com"];
```

The origin that you configure in the `ThirdPartyMap.js`'s `allowedOrigins` must be identical to the viewer's origin as it appears in the browser. Note that `http://myviewers` does not match `http://myviewers.mydomain.com`, even though they map to the same machine.

5. Save the file.

## Configure Access to a Secured Site by an Integrated Application

If the site that the viewer belongs to is secured and SSL is enforced (in other words, the Enforce SSL setting is enabled in the Security Settings), then you must ensure that the web page uses SSL to access the integrated application.

At a minimum, you must specify `https` in the URL that the web page uses to connect to the application. Depending on the application, you may also have to include a URL parameter for SSL. Refer to the application's documentation for information about secure access.

### Example 1: Integrate with Bing Maps

This example shows how to integrate Bing Maps into an HTML5 viewer using the sample Bing Maps web page that ships with the Viewer.

In this example, the integration pane (`ExternalComponentView`) is closed by default and the toolbar includes the **Linked Maps** button. When the user clicks the button, Bing Maps opens in an integration pane with a header.

The main steps to adapt the Bing Maps sample are:

1. Adapt the sample Bing Maps web page:
  - a. Configure your Bing Maps key.
  - b. If the viewer belongs to a secured site and SSL is enforced, adapt the Bing Maps URL to use SSL.
2. Add an external component for Bing Maps to the Integration Module.
3. Configure cross-domain access.

You only need to do this if the sample Bing Maps web page is deployed to a different domain than the viewer.
4. Add the Linked Maps button to the toolbar.

The instructions assume that the Bing Maps web page is hosted at `http://mymaps.mydomain.com/maps` and the HTML5 Viewer is hosted at `http://myviewers.mydomain.com/Html5Viewer`. Replace these URLs with the URLs for your installation.

### Step 1: Adapt the sample Bing Maps web page

1. Run an HTML editor or text editor as an administrator.
2. Locate `BingMaps.html` where it is deployed in IIS.

In the default deployment, the file is here:

```
C:\inetpub\wwwroot\Html5Viewer\Resources\3rdPartyMaps\BingMaps.html
```

3. Open `BingMaps.html` in the editor.
4. Configure your Bing Maps key:
  - a. Find the following line:

```
var bingApiKey = "";
```

- b. Type or paste your Bing Maps key between the quotation marks.
5. Save the file.

## Step 2: Add an external component for Bing Maps to the Integration Module

If you already have one or more `externalComponents` configured when you add the Bing Maps external component, remember to separate the components using commas. You may also want to configure the `defaultComponent` if you have more than one external component configured.

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js` or `Tablet.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST  
Elements\Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. Add an external component for Bing Maps to the Integration Module's `externalComponents` property:

```
{
  "moduleName": "Integration",
  ...
  "viewModels": [
    {
      ...
      "configuration": {
        ...
        "externalComponents": [
          {
            "id": "bingMaps",
            "displayName": "Bing Maps",
            "uri": "Resources/3rdPartyMaps/BingMaps.html",
            "viewpointIndicatorUri": "Resources/Images/Icons/location-direction-
red-32.png"
          }
        ]
      }
    }
  ]
  ...
}
```



The example markup given above assumes that the web page and the viewer are deployed to the same domain. If they are deployed to different domains, you must specify the full URL, for example, `http://mymaps.mydomain.com/maps/Resources/3rdPartyMaps/BingMaps.html`.

4. Save the file.
5. Repeat these steps for the other configuration file.

### Step 3: Configure cross-domain access

If the Bing Maps web page is deployed to a different domain than the viewer, then you must configure cross-domain access so the viewer and integration pane can communicate with each other. If the web page and viewer are deployed to the same domain, skip this step.

1. Configure the web page's origin:
  - a. In one of the viewer's configuration files, add the web page's origin to the Integration Module's `allowedOrigins` property:

```
{
  "moduleName": "Integration",
  "moduleType":
"geocortex.essentialHtmlViewer.mapping.modules.integration.IntegrationModule",
  "configuration": {
    "allowedOrigins": ["http://mymaps.mydomain.com"]
  }
  ...
}
```



The origin that you configure in the `allowedOrigins` property must be identical to the URL that you configure in the `externalComponent's uri` property.

- b. Save the file.
  - c. Repeat these steps for the other configuration file.
2. Configure the viewer's origin:
  - a. Open `ThirdPartyMap.js`.  
In the default deployment, the file is here:

```
C:\inetpub\wwwroot\Html5Viewer\Resources\3rdPartyMaps\ThirdPartyMap.js
```

- b. Find the following line:

```
this.allowedOrigins = [];
```

- c. Type or paste the viewer's origin in the array, within quotation marks:

```
this.allowedOrigins = ["http://myviewers.mydomain.com"];
```

The origin that you configure in `ThirdPartyMap.js's allowedOrigins` property must be identical to the viewer's origin as it appears in the browser. Note that `http://myviewers` does not match `http://myviewers.mydomain.com`, even though they map to the same machine.

- d. Save the file.

## Step 4: Add the Linked Maps button to the toolbar

1. In Manager, edit the viewer.
2. Click **Toolbar** in the side panel.
3. In the **Available Tools** column, find the **Linked Maps** tool and drag it to the desired position in the **Configured**

**Toolbar** column.

4. Click **Apply Changes**.
5. Click **Save Site**.

## Example 2: Integrate with Pictometry

This example shows how to integrate Pictometry into an HTML5 viewer using the sample Pictometry web page that ships with the Viewer.

In this example, the integration pane (`ExternalComponentView`) is closed by default and the toolbar includes the **Linked Maps** button. When the user clicks the button, Pictometry opens in an integration pane with a header.

The main steps to adapt the Pictometry sample are:

1. Adapt the sample Pictometry web page:
  - a. Configure your Pictometry keys.
  - b. If the viewer belongs to a secured site and SSL is enforced, adapt the Pictometry URLs to use SSL.
2. Add an external component for Pictometry to the Integration Module.
3. Configure cross-domain access.
 

You only need to do this if the sample Pictometry web page is deployed to a different domain than the viewer.
4. Add the Linked Maps button to the toolbar.

### Step 1: Adapt the sample Pictometry web page

1. Run a text editor as an administrator.
2. Locate `Pictometry.aspx` in the viewer's `3rdPartyMaps` folder in IIS.

In the default deployment, the file is here:

```
C:\inetpub\wwwroot\Html5Viewer\Resources\3rdPartyMaps\Pictometry.aspx
```

3. Open `Pictometry.aspx` in the editor.
4. Configure your Pictometry keys:
  - a. Find the following lines:

```
protected static string ApiKey = "";
protected static string SecretKey = "";
```

- b. Type or paste your Pictometry keys:
    - i. Put the API Key between the `ApiKey` quotation marks.
    - ii. Put the Secret Key between the `SecretKey` quotation marks.

The API Key and Secret Key are provided by Pictometry.

5. If the viewer belongs to a secured site and SSL is enforced:

- a. Find the following lines:

```
public string IpaLoadUrl = "http://pol.pictometry.com/ipa/v1/load.php";  
public string IpaJsLibUrl =  
"http://pol.pictometry.com/ipa/v1/embed/host.php?apikey=" + ApiKey;
```

- b. Change the protocols to https:

```
public string IpaLoadUrl = "https://pol.pictometry.com/ipa/v1/load.php";  
public string IpaJsLibUrl =  
"https://pol.pictometry.com/ipa/v1/embed/host.php?apikey=" + ApiKey;
```

6. Save the file.

## Step 2: Add an external component for Pictometry to the Integration Module

If you already have one or more `externalComponents` configured when you add the Pictometry external component, remember to separate the components using commas. You may also want to configure the `defaultComponent` if you have more than one external component configured.

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js` or `Tablet.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST  
Elements\Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default\
```



3. Add an external component for Pictometry to the Integration Module's `externalComponents` property:

```
{
  "moduleName": "Integration",
  ...
  "viewModels": [
    {
      ...
      "configuration": {
        ...
        "externalComponents": [
          {
            "id": "pictometry",
            "displayName": "Pictometry",
            "uri": "Resources/3rdPartyMaps/Pictometry.aspx",
            "viewpointIndicatorUri": "Resources/Images/Icons/location-direction-
purple-32.png"
          }
        ]
      }
    }
  ]
  ...
}
```



The example markup given above assumes that the web page and the viewer are deployed to the same domain. If they are deployed to different domains, you must specify the full URL, for example, `http://mydomain.com/maps/Resources/3rdPartyMaps/Pictometry.aspx`.

4. Save the file.
5. Repeat these steps for the other configuration file.

### Step 3: Configure cross-domain access

If the Pictometry web page is deployed to a different domain than the viewer, then you must configure cross-domain access so the viewer and integration pane can communicate with each other. If the web page and viewer are deployed to the same domain, skip this step.

1. Configure the web page's origin:
  - a. In one of the viewer's configuration files, add the web page's origin to the Integration Module's `allowedOrigins` property:

```
{
  "moduleName": "Integration",
  "moduleType":
"geocortex.essentialsHtmlViewer.mapping.modules.integration.IntegrationModule",
  "configuration": {
    "allowedOrigins": ["http://mymaps.mydomain.com"]
  }
  ...
}
```



The origin that you configure in the `allowedOrigins` property must be identical to the URL that you configure in the `externalComponent's uri` property.

- b. Save the file.
  - c. Repeat these steps for the other configuration file.
2. Configure the viewer's origin:
  - a. Open `ThirdPartyMap.js`.  
In the default deployment, the file is here:

```
C:\inetpub\wwwroot\Html5Viewer\Resources\3rdPartyMaps\ThirdPartyMap.js
```

- b. Find the following line:

```
this.allowedOrigins = [];
```

- c. Type or paste the viewer's origin in the array, within quotation marks:

```
this.allowedOrigins = ["http://myviewers.mydomain.com"];
```

The origin that you configure in the `ThirdPartyMap.js` `allowedOrigins` must be identical to the viewer's origin as it appears in the browser. Note that `http://myviewers` does not match `http://myviewers.mydomain.com`, even though they map to the same machine.

- d. Save the file.

## Step 4: Add the Linked Maps button to the toolbar

1. In Manager, edit the viewer.
2. Click **Toolbar** in the side panel.
3. In the **Available Tools** column, find the **Linked Maps** tool and drag it to the desired position in the **Configured**

**Toolbar** column.

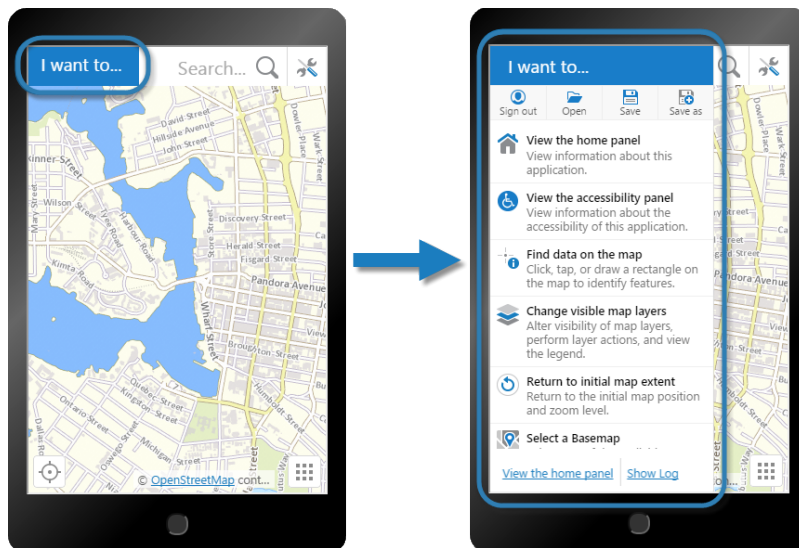
4. Click **Apply Changes**.
5. Click **Save Site**.

## 15.33 IWantToMenu Module



This module can be configured using Manager. For instructions, see [Configure the I Want To Menu on page 79](#).

The I Want To menu is a list of frequently performed tasks. The IWantToMenu Module implements the I Want To menu and its menu items. The menu is called the I Want To menu because the default text on the button that opens the menu is **I want to...**. In the Desktop and Tablet interfaces, the I Want To button is in the top left corner of the map. In the Handheld interface, the button is at the top left of the screen.



**I Want To menu shown in the Handheld preview**

If you do not want the viewer to have an I Want To menu, you can hide it by clearing the **Show Menu** checkbox on the viewer's I Want To Menu page in Manager. Alternatively, you can set the `IWantToMenuViewModel`'s `showMenu` property to `false` in the configuration files.

## Global Menu

By default, the I Want To menu has a horizontal menu at the top, which is implemented by the GlobalMenu Module. You can hide the Global menu using the `IWantToMenuViewModel`'s `showGlobalMenu` property. For example, you might want to hide the Global menu if the viewer does not allow users to sign in. For more information about the Global menu, see [GlobalMenu Module on page 187](#).

## Configuration Properties

### Module

- **menus:** An array of menus with one menu in it—the I Want To menu. The menu has the following properties:
  - **id:** The menu's ID.
  - **description:** A short description of the menu.



If your viewer is going to be available in more than one language, enter the text key that the description is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **moduleId:** The ID of the module that the menu belongs to.
- **defaultIconUri:** The URI of the default icon for menu items.
- **items:** An array of menu items. Menu items have the following properties:
  - **iconUri:** The image to display beside the menu item. The recommended image size is 24px by 24px.
  - **text:** The text to appear on the menu item. You can use a text key or the literal text.
  - **description:** A brief description of the menu item to appear below the `text`. You can use a text key or the literal text.
  - **command:** The command to execute when the menu item is selected.



If you set the `command` property, do not set the `batch` property.

- **commandParameter:** The parameter value to pass to the command when it runs, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.



If you set the `commandParameter` property, do not set the `batch` property.

In the following example from the **FeatureActions** menu, a **commandParameter** is passed into the **ShowFeatureDetailsCompact** command to display feature details in the sidebar:

```
"command": "ShowFeatureDetailsCompact",
"commandParameter": "{{context}}"
```

The **{{context}}** token is a special token that retrieves the default menu context which a menu may have. For example, the default context of the `FeatureActions` menu is the current `feature`. Therefore, the command displays feature details about the current `feature`.

- **hideOnDisable:** If this property is set to `true` and the menu item's command cannot run, the

menu item does not show in the menu.

If `hideOnDisable` is `false` and the menu item's command cannot run, the menu item shows in the menu, but it is grayed out.

- **batch**: An array of objects, each with three properties: an executable command, an optional command parameter, and an optional Boolean to abort the execution of further commands if a command fails to execute. This allows multiple commands to be executed in a specified order, although some commands may trigger asynchronous actions.

The following example attempts to activate the Tabbed Toolbar and, if successful, proceeds to greet the user with an alert:

```
"batch": [
  {
    "command": "ActivateView",
    "commandParameter": "TabbedToolbarView",
    "abortBatchOnFailure": true
  },
  {
    "command": "Alert",
    "commandParameter": "Hello!"
  }
]
```

If omitted, `abortBatchOnFailure` is `false` by default.



If you set the `batch` property, do not set the `command` or `commandParameter` properties.

## Views

- **IWantToMenuButtonView**: (Desktop and Tablet interfaces) None
- **IWantToMenuView**:
  - `menuId`: The ID of the menu displayed in the view.

## View Models

- **IWantToMenuViewModel**:
  - `showMenu`: When this property is set to `true`, the I Want To menu shows in the viewer. If `showMenu` is set to `false`, the viewer does not have an I Want To menu. The default is `true`.
  - `showGlobalMenu`: When this property is set to `true`, the Global menu shows in the I Want To menu. If `showGlobalMenu` is set to `false`, the I Want To menu does not have a Global menu. The default is `true`.

- menuTitle:** The text that appears on the button that opens the I Want To menu, by default, **I want to...**  
 If your viewer is going to be available in one language only, type the menu title. If your viewer is going to be available in more than one language, enter the text key that the menu title is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.
- primaryButtonColor:** The color of the I Want To menu button as a RRGGBB hex code. The default value is #1A72C4.

### See Also...

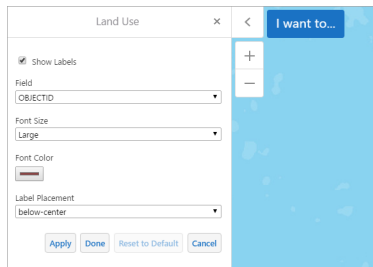
[GlobalMenu Module on page 187](#)

[Menu Module on page 282](#)

## 15.34 LabelOptions Module

The LabelOptions Module configures feature label fonts and styles for dynamic layers-enabled map services. It also provides default label styles if the layer does not already have set styles. For more information about configuring symbolization, see "Symbolization" in the *Geocortex Essentials Administrator Guide*.

By default, the viewer allows users to customize labels's font color, font size, placement, and the layer field to display. You can configure additional label options to be configured in the viewer from the viewer configuration files. See [Configuration Properties](#) for information.



The Customize Label Options panel for a dynamic layer called Land Use

### ▶ To customize label options in the viewer:

1. Locate the layer you want to customize from the layer list.
2. Select the ▶ Action button next to the layer name.
3. Select the **Customize Labels** action.  
Alternatively, you can quickly toggle the feature labels by selecting the **Toggle Labels** action.
4. If label styles are already defined, you must choose the **Customize** button to begin customizing them. If label styles are not defined, you are presented with forms and drop-down menus to modify the label styles. For more information see [Default Layer Styles on page 238](#).
5. Change the label styles using the available forms and drop-down menus.
6. Choose **Apply** to apply the new styles.

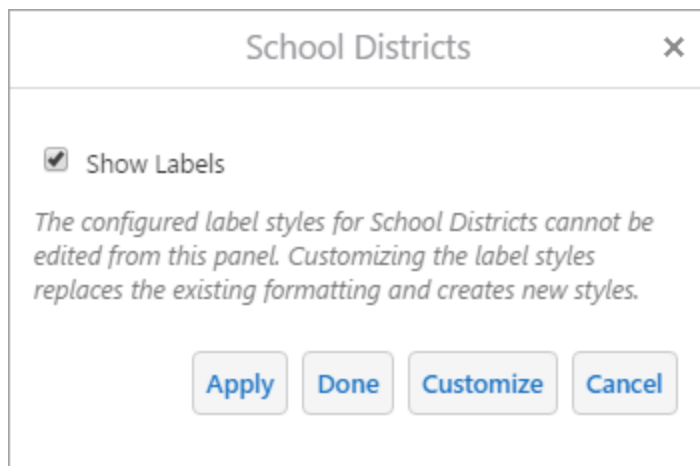
You can choose **Reset to Default** to clear the changes that you have made.

7. Choose **Done** or exit the Label Options panel.

## Default Label Styles

If no label styles have been defined previously (either in the layer properties or in the dynamic definitions set in Essentials), when the user chooses to customize the labels the style values default to the styles set in the viewer's configuration files. If the user customizes the styles, they cannot revert the styles to their original state for the rest of the session.

Before users can access the label options panel, they are informed that the configured styles cannot be edited, and that customizing the styles replaces the existing styles. The user can use the Customize button to continue modifying the styles.



This dialog means that no label styles have been defined previously

## Modify Label Options

Users can modify label styles using the following options:

- **Show Labels:** When this checkbox is selected, a label is displayed for each feature on the map. Alternatively, you can use the Toggle Labels layer action to quickly show or hide all of the labels.
- **Field:** Chooses the field that is used as the label. Eligible fields from the layer are displayed in a drop-down menu.
- **Font Size:** Chooses the font size for the labels from a drop-down menu. The default values enabled are Extra Small (10), Small (12), Medium (14), Large (16), and Extra Large (18). However, if `fontSizeChoiceIsEnabled` is set to `false` in the viewer configuration files, then the user can input a numeric value for the font size instead. See [Configuration Properties](#) for information.
- **Font Family:** Chooses the font family for the labels from a drop-down menu. By default, the available font families are Arial, Cambria, Georgia, Times New Roman, and Verdana. You can configure additional fonts in the viewer configuration files. See [Configuration Properties](#) for information.
- **Font Style:** Select whether labels are bold, italic, and underlined. The Font Style configuration checkboxes do not appear in the viewer by default. They must first be configured in the viewer configuration files. See [Configuration Properties](#) for information.

- **Bold:** Select this checkbox to display the labels as bold text.
- **Italic:** Select this checkbox to display the labels as italic text.
- **Underline:** Select this checkbox to display the labels as underlined text.
- **Font Color:** Choose a font color using the color picker tool.
- **Label Placement:** Choose the placement of the label relative to the feature.  
Label placement is configured differently for point, line, and polygon features. In the case of polygons, label placement cannot be configured at all.
- **Points:** Labels for point features have nine configurable placements. All the placements are either horizontally left, at the center, or right of and vertically above, at the center of, or below the feature.

above-left   above-center   above-right  
center-left   center-center   center-right  
below-left   below-center   below-right

The nine configurable Label Placement options for point geometries.

- **Lines:** Labels for line features have fifteen configurable placements. All of the placements are either horizontally before, at the start of, along, at the end of, or after the feature, and vertically above, at the center of, or below the feature.

above-before   above-start   above-along   above-end   above-after  
center-before   center-start   center-along   center-end   center-after  
below-before   below-start   below-along   below-end   below-after

The fifteen configurable Label Placement options for point geometries.

- **Polygons:** Label placement for polygons cannot be set.
- **Outline Width:** The width of the outline around the label in pixels.  
The Outline Width input field does not appear in the viewer by default. It must first be configured in the viewer configuration files. See [Configuration Properties](#) for information.
- **Outline Color:** The color of the outline around the label.  
The Outline Color input field does not appear in the viewer by default. It must first be configured in the viewer configuration files. See [Configuration Properties](#) for information.



---

## Configuration Properties

### Module

None

### Views

- **LabelOptionsView:** None

### View Models

- **LabelOptionsViewModel:**
  - **showLabels:** The default value is `true`.
  - **fontSizeExtraSmall:** Defines the Extra Small preset font size. The default value is 10.
  - **fontSizeSmall:** Defines the Small preset font size. The default value is 12.
  - **fontSizeMedium:** Defines the Medium preset font size. The default value is 14.
  - **fontSizeLarge:** Defines the Large preset font size. The default value is 16.
  - **fontSizeExtraLarge:** Defines the Extra Large preset font size. The default value is 18.
  - **fontSizeChoiceIsEnabled:** Enables the preset values for Extra Small, Small, Medium, Large, and Extra Large font sizes as a drop-down menu. When this setting is disabled, the user can input the font size as a numeric value. The default value is `true`.
  - **fontSize:** The default font size for unstyled labels. The default value is 14.
  - **fontColor:** The default font color for unstyled labels. The default value is `#000000`.
  - **fontFamilies:** Configures the list of available font families that can be used for labels. The default value is "Arial", "Cambria", "Georgia", "Times New Roman", "Verdana". Add any web-safe font to this array.
  - **fontIsBold:** Enables a checkbox that allows users to make label fonts bold. The default value is `false`.
  - **fontIsItalic:** Enables a checkbox that allows users to make label fonts italic. The default value is `false`.
  - **fontIsUnderline:** Enables a checkbox that allows users to make label fonts underlined. The default value is `false`.
  - **haloSize:** The default width of an unstyled label's outline in pixels. The default value is 0.
  - **haloColor:** The default color of an unstyled label's outline in pixels. The default value is `#000000`.
  - **showLabelsIsVisible:** Enables a checkbox that allows users to toggle labels on or off. The default value is `true`.
  - **fieldIsVisible:** Enables the Field drop-down menu. The user can configure which layer field is displayed as the label. The default value is `true`.
  - **fontSizeIsVisible:** Enables the Font Size option. The default value is `true`.

- **fontColorIsVisible:** Enables the Font Color option. The default value is `true`.
- **labelPlacementIsVisible:** Enables the Label Placement option. The user can configure where the label should appear in relation to the feature. The default value is `true`.
- **fontFamiliesIsVisible:** Enables the Font Family picker. The default value is `false`.
- **fontStyleIsVisible:** Enables the Font Style options, which includes Bold, Italic, and Underline checkboxes. The default value is `false`.
- **haloSizeIsVisible:** Enables the Outline Width option. The default value is `false`.
- **haloColorIsVisible:** Enables the Outline Color option. The default value is `false`.

## 15.35 LayerAddition Module

The LayerAddition Module allows end users to search for layers and map services that have been configured as service connections within Essentials Manager.

See "Enable Searching a Service Connection from Viewers" in the *Essentials Administrator Guide* for more information about how to enable searching service connections from the viewer.

### Configure the Add Layers Tool

To allow users to add layers and services from configured service connections, you can configure the Add Layers tool to the viewer toolbar. For more information about configuration toolbar items, see [Configure the Toolbar on page 108](#).



#### The Add Layers tool

Alternatively, you can activate the Add Layers user interface with the `AddMapLayerInteractive` command.

### Add Layers User Interface

When the Add Layers function is activated, the Add Layers user interface appears in the viewer's main panel. Adding layers and map services is a three step process.

#### ► To add a layer or map service to the layer list:


1. Use the **Add Layers** tool in the toolbar to activate the Add Layers user interface.
2. Enter a search term or the URL of a known map service and press the **Search** button.

The Search for Layers dialog

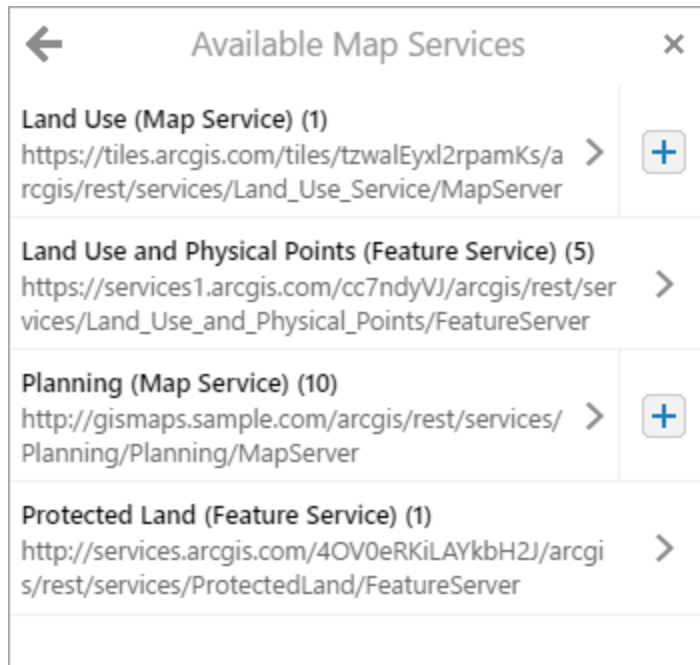
3. Select a configured service connection.

The number of relevant search results for each connection is displayed in parentheses next to the service connection's name. For example, "ArcGIS Online (20)". If a service connection returned no results for the search term, the service connection does not appear in this list.

The Search Connections dialog

4. Select a result from the list of available map services.
  - Use the  Add button next to a map service or layer to add it to the Layer List.
  - Select a result to see a list of available layers that you can add.

The number of available layers is displayed next to the service's name. For example, "Wells (Feature Service) (6)".




#### The Available Map Services dialog


The Add Layers user interface closes.

The user-added layer appears at the top of the layer list, unless you have specified that user-added layers should appear within a configured folder. For more information, see "The Layer List" in the *Essentials Administrator Guide*.

## User-Added Layer Actions

You can configure the  Action Menu associated with layers and user-added layers from the Management Pack or the viewer configuration files. With the viewer's default configuration, the Action Menu for user-added layers includes the Edit Layer Properties action, the Remove Layer action, and all of your other configured layer actions. Only dynamically added layers can access the Edit Layer Properties and Remove Layer actions. For more information about menu configuration, see [Configure the Context Menus on page 81](#) or [Menu Module on page 282](#).

### Edit Layer Properties

When the user selects the  Action Menu button next to a user-added layer, they can select the **Edit Layer Properties** action if it is configured. This action allows users to change the way that the user-added-layer behaves in the viewer. The following configuration settings are available:


- **Name:** The name of the user-added layer in the Layer List.
- **Identify enabled:** When this checkbox is selected, identify operations include results from the user-added layer that is being configured.
- **Search enabled:** When this checkbox is selected, search operations include results from the user-added layer that is being configured.

- **Query enabled:** When this checkbox is selected, query operations included results from the user-added layer that is being configured.
- **Show map tips:** When this checkbox is selected, map tips appear when the user-added layer's features are selected.



Group layers are not searchable or identifiable. As a result, group layers do not include the **Identify enabled**, **Search enabled**, or **Query enabled** checkboxes in the Edit Layer Properties dialog.

## User-Added Map Service Actions

You can configure the  Action Menu associated with user-added map services from the viewer configuration files. By default, the only configured menu item is the Remove Service action, which removes the user-added map services from the Layer List. Only dynamically added map services can access the Remove Service action. For more information about menu configuration, see [Configure the Context Menus on page 81](#) or [Menu Module on page 282](#).

## Configuration Properties

### Module

None

### Views

- **AddLayerDialogView:** None
- **ServiceConnectionsDialogView:** None
- **MapServicesDialogView:** None
- **SubLayersDialogView:** None
- **LayerPropertiesView:** None

### View Models

- **LayerAdditionViewModel:**
  - **zoomToUserAddedLayers:** When a layer is added, the map zooms to its full extent. The default value is `true`.
  - **layerDefaults:**
    - **searchable:** Allows user-added layers to participate in Global Search. The default value is `true`.
    - **identifiable:** Allows user-added layers to be used with identify operations. The default value is `true`.
    - **queryable:** Allows user-added layers to be used with the Query Builder. The default value is `true`.
    - **showMapTips:** Displays map tips for all user-added layers. The default value is `true`.

- `LayerPropertiesViewModel: None`

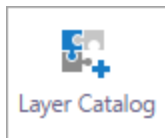
## 15.36 LayerCatalog Module

The LayerCatalog Module enables users to choose layers from a referenced layer catalog site. The layer catalog user interface can be used to add additional layers and map services to the map.

When a user adds a layer from a layer catalog to the map, it appears in a Layer Catalog folder in the layer list. If, in Essentials Manager, you have configured a folder where user-added layers should appear in the layer list, the Layer Catalog folder is placed within that configured folder. For more information about configuring this folder, see "The Layer List" in the *Essentials Administrator Guide*.

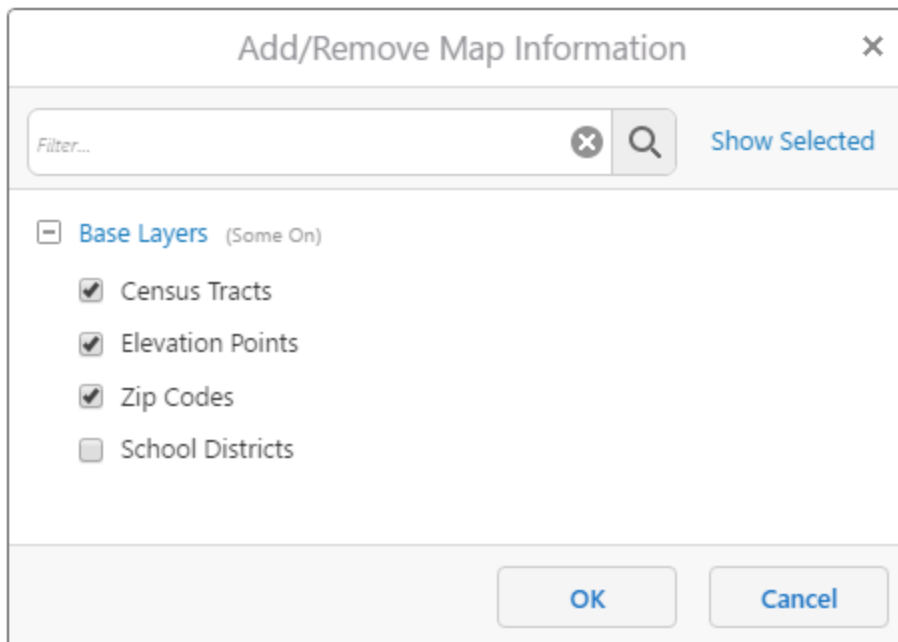
Note that with a layer catalog, you cannot reference the same map service that is already present in the base site. Also, layer catalog layers do not support attachments, data links, instant search, and relationships.

You can configure the Layer Catalog tool to allow users to activate the Layer Catalog user interface. For toolbar configuration instructions, see [Configure the Toolbar on page 108](#).



### The Layer Catalog tool

Alternatively, use the `ShowLayerCatalog` command to invoke the modal window where users can make a selection.



The Add/Remove Map Information modal window

## Configure a Layer Catalog Site

In order to access a layer catalog, your site must reference a layer catalog site unless you use an assembly provider. You can configure layer catalog sites in Essentials Manager. For more information, see "Layer Catalog Sites" in the *Essentials Administrator Guide*.

## Legacy Layer Catalogs and Assembly Providers



The Legacy Layer Catalogs feature requires custom development to your Essentials installation and has been deprecated in favor of layer catalog sites. Legacy layer catalogs only work with dynamic layers.

If you wish to use an assembly provider instead of a layer catalog site, you can disable the default `layerCatalogProviders` property in your viewer configuration files and enable an assembly provider. Using an assembly provider requires additional Essentials configuration. See "Legacy Layer Catalogs" in the *Essentials Administrator Guide* for more information.

## Configuration Properties

### Module

- **layerCatalogProviders:** Configure additional layer catalog providers and enable or disable them. Only one layer catalog provider should be enabled.
  - **type:** A provider. The default enabled provider is:
 

```
"geocortex.essentialsHtmlViewer.mapping.modules.layerCatalog.EssentialsSiteProvider".
```

A default assembly provider is also provided, but is disabled by default:

```
"geocortex.essentialsHtmlViewer.mapping.modules.layerCatalog.AssemblyProvider".
```
  - **enabled:** Use the value `true` to enable the configured provider `type` above.
- **checkboxes:** Configure the checkboxes next to each folder of layers in the layer catalog.
  - **visibility:** Configure the visibility of the checkboxes next to each folder of layers. You can configure this property to `all`, `none`, or `configuredOnly`.
  - **suffix:** If the `visibility` property is set to `configuredOnly`, you can configure a suffix which folders of layers are configured with checkboxes. For example, you could configure a suffix `_GROUP`.

### Views

- **LayerCatalogView:** The Add/Remove Map Information modal window. This view allows users to select or deselect layers and map services they wish to display in the map's Layers panel.


- **autoCompleteEnabled**: If enabled, the filter bar suggests matching entries to the user as they enter a query into the filter bar. The default value is `true`.
- **minFilterLength**: Sets the minimum length of characters that a filter query can be. The default value is 3.

## View Models

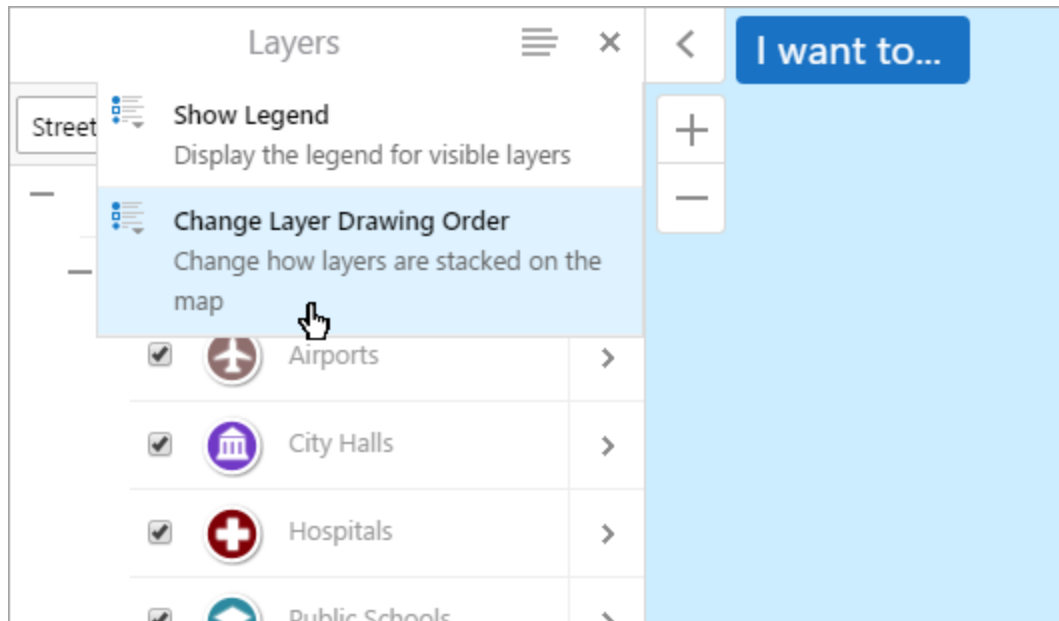
- **LayerCatalogViewModel**:
  - **infoLink**: An object whose properties configure an info link for all layer catalog items. This property is omitted from the default configuration files. By default, info links are not available in the layer catalog user interface.
    - **enabled**: Enables or disables the info link for all layer catalog items. The default value is `false`.
    - **command**: The command to run when the user chooses a layer's info link.
    - **iconUri**: The URI to an icon that is displayed next to an available layer in the layer catalog. The default value is `Resources/Images/Icons/info-12.png`.
    - **tooltip**: The tool tip to display when the info link is hovered over.
    - **text**: The info link's text label. This property is only displayed if the `iconUri` is not configured.

## 15.37 LayerDrawingOrder Module

Users can modify the drawing order of map services and layers. This is especially useful when a site includes a large amount of graphical layers, or when the user needs to add their own layers using a layer catalog or the Add Layers tool.

In desktop shells, users can activate the drawing order user interface from the layer list's  Panel Actions Menu using the **Change Layer Drawing Order** action. On handheld shells, this action appears in the I Want To menu.





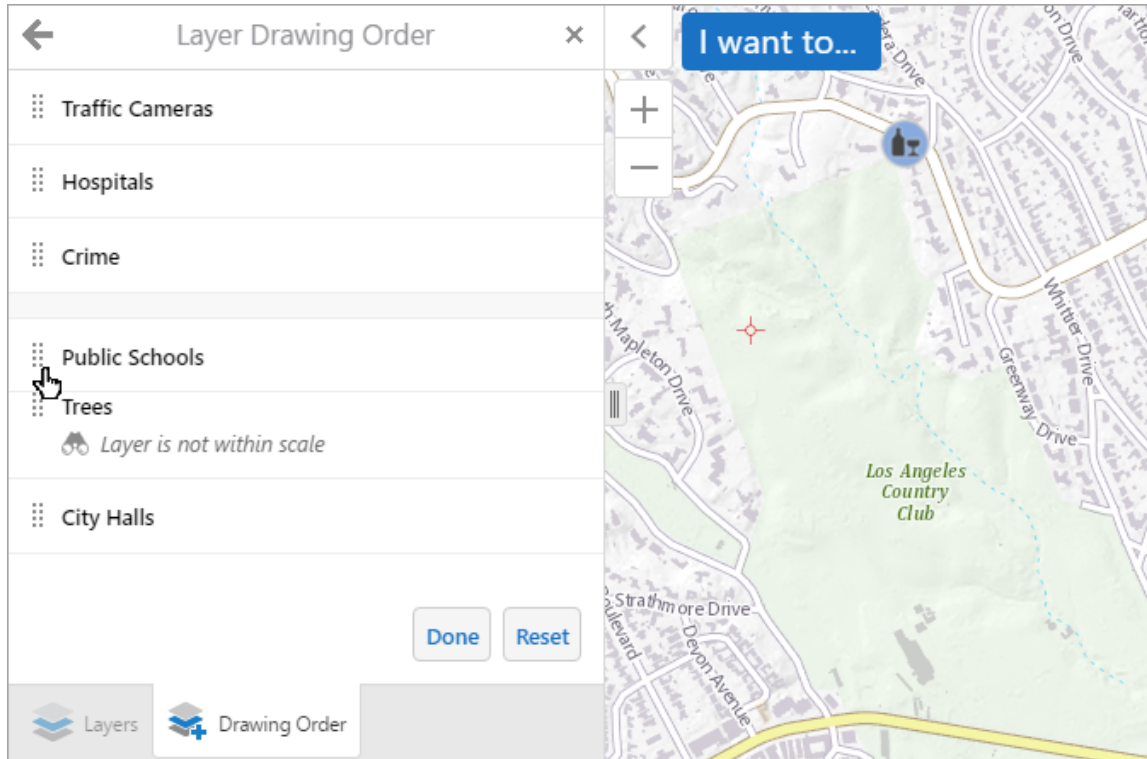
The Change Layer Drawing Order action selected in the layer list's Panel Actions Menu

## User Interface

Once the panel is open, users can choose to change the layer drawing order for **Graphical Layers** or **Map Services**. Both of these options lead to the main drawing order user interface, where items in the drawing order appear from top to bottom and can be rearranged using the drag handle on the left side of each item.

If users want to undo their changes, they can use the **Reset** button to reset the drawing order back to when it was last-refreshed. (Note that the panel is refreshed when the user closes and reopens the Layer Drawing Order panel, or when a new layer is added or removed from the map.)

Users can close the panel or use the **Done** button to save the new state and continue using the viewer.



A layer being moved down in the drawing order in the main layer drawing order user interface


### Drawing Order Keyboard Accessibility

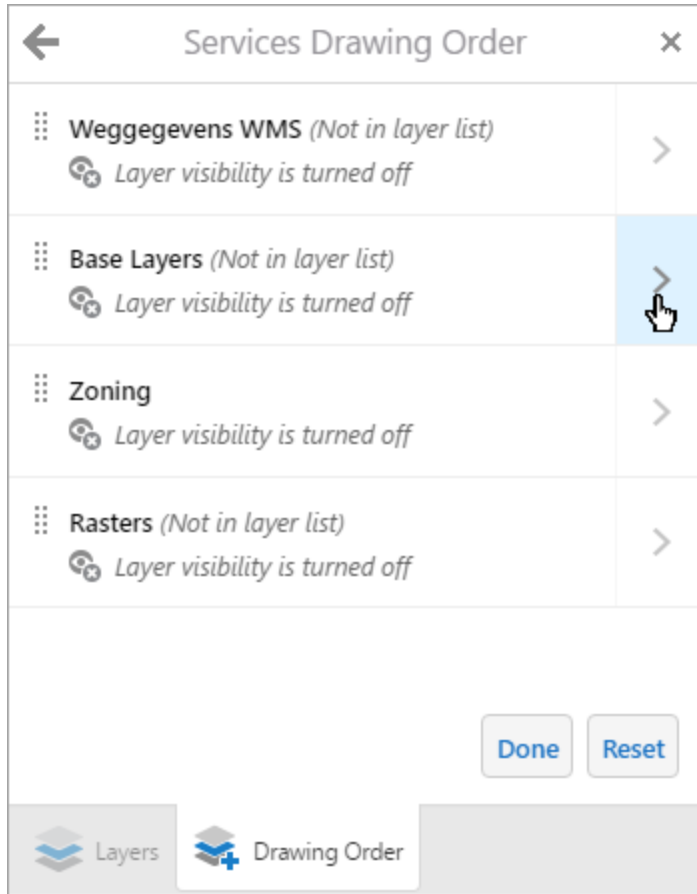
If the user is unable to click and drag items in the drawing order to rearrange them, they can click or tap an item to reveal arrows that move the item up or down one position in the list.



Items in the drawing order can be clicked or pressed to reveal arrows in the case that users cannot drag and drop

### Dynamic Map Service Layer Order

ArcGIS Dynamic Map Services can be reordered in the list of map services, and when you select the  action button, you can also reorder the layers contained within the map service.



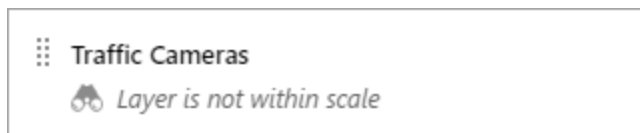
The Services Drawing Order panel may include embedded layer order lists for dynamic map services

## Drawing Order Meta Information

If layers are not in the layer list, are not visible, or are not in scale, the user interface appends meta information to the map service or layer at hand as descriptor text. In some cases, an item in the drawing order may display multiple meta information descriptors.

Meta information about the layer's visibility on the map can appear below an item in the drawing order list:

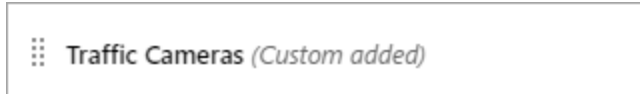
- **Layer is not within scale:** Based on the current map scale, the layer does not appear viewable on the map.
- **Layer visibility is turned off:** The layer cannot be viewed on the map because its layer visibility is set to off. You can change layer visibility for layers and groups of layers from the layer list.



Meta information that gives the user more information about a layer's visibility

Additional meta information about the layer itself may also appear next to the label:

- **(Custom added):** The layer has been added by the user.
- **(Not in layer list):** The layer is not displayed in the layer list, but it is visible on the map unless a second descriptor is explicitly states that it is not.



Meta information that gives the user more information about a layer

## Basemaps and Markup Layers

Basemaps and markup layers cannot be reordered, and they are not displayed in the drawing order.

Basemaps cover any layers configured below them in the drawing order, and administrators should configure them in Essentials Manager to be drawn first, at the bottom of the drawing order. For information about configuring the drawing order in Essentials, see the topic "Change the Map's Drawing Order" in the *Essentials Administrator Guide*.

Markup layers (such as user drawings, measurements, and plotted coordinates) are always drawn on top of any items in the drawing order.

## Configuration Properties

Module

None

Views

- **LayerGroupsDialogView:** None
- **OrderServicesDialogView:** None
- **OrderLayersDialogView:** None

View Models

- **LayerDrawingOrderViewModel:** None

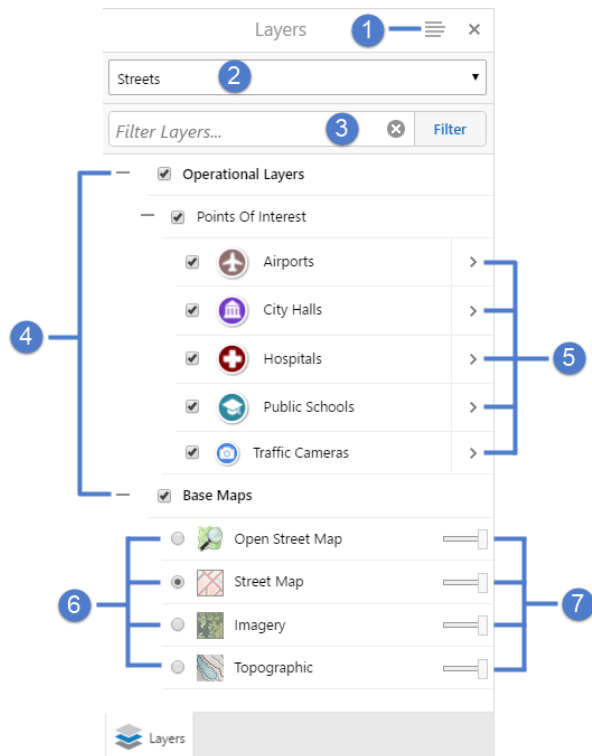
## 15.38 LayerList Module



This module can be configured using Manager. For instructions, see [Configure the Layer List on page 103](#).

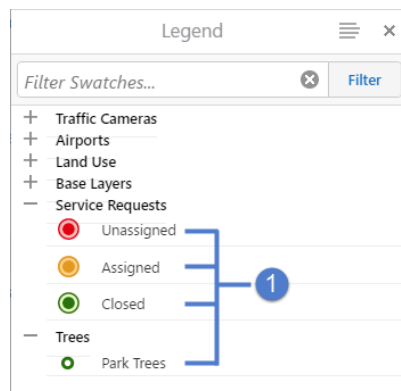
The LayerList Module controls the Layers panel that displays on the left side of the viewer by default. It contains the configurable directory of folders, map services, layers and basemaps that are available in the viewer. End users use the Layers panel to change the visibility of layers, show legend swatches, change the transparency of layers, and more.

You can use Manager to control which folders, map services and layers are initially turned on in the Layers panel when a user launches the map. For information, see *The Layer List* in the *Geocortex Essentials Administrator Guide*.



### Layers panel

1	Panel Actions menu	2	Layer Theme selection	3	Layer List Filter input field
4	Folders	5	Opens Layer Actions menus	6	Visibility settings
7	Transparency sliders				



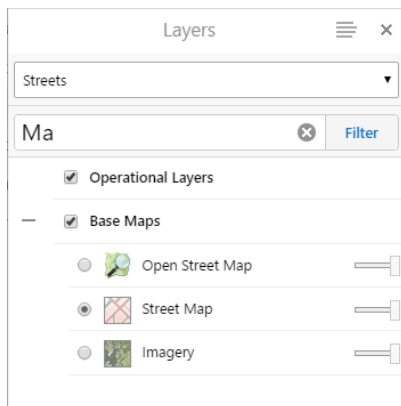
### Legend panel

1	Legend swatches
---	-----------------

## Filter the Layer List

When the user begins typing in the **Filter** input field, the layer list shows the layer names that match or contain the text in the **Filter** field. If the matching layers are in folders, those folders are displayed and expanded to show the layer names.

For example, the following image shows the results of filtering the layer list to show any layers containing "Ma". The search is not case sensitive, and in this example **Imagery** is included in the results because it contains "ma".



If the user selects **Show Legend** in the Panel Actions menu, the **Filter** input field indicates that the filtering applies to swatches.

## Configuration Properties

### Module

- **enableLegendIntegration**: To enable the ability to display legend swatches within the layer list, set to `true`; otherwise, set to `false`. The default is `true`.
- **onlyShowSwatchesOnVisibleLayers**: To only display legend swatches for layers that are visible, set to `true`; otherwise, set to `false`. The default is `false`.
- **autoActivateAncestorVisibilities**: To automatically toggle the visibilities of parent items when the visibility of an item in the layer list is changed, set to `true`; otherwise, set to `false`. The default is `false`.
- **enableLayerIcons**: To display layer icons for each layer in the layer list that has layer icons configured and does not have a legend, set to `true`; otherwise, set to `false`. The default is `false`.

### Views

- **LayerListView**: None
- **LayerActionsView**:
  - **menuId**: The ID of the menu of layer actions that is configured in the [Menu Module](#).
- **MapServiceActionsView**:
  - **menuId**: The ID of the menu of map service actions that is configured in the [Menu Module](#).

## View Models

- **LayerListViewModel:**
  - **showTransparencySlider:** To display transparency sliders for each map service, set to `true`; otherwise set to `false`. The default is `true`.
  - **autoExpandLegendSwatches:** To automatically display the legend swatches for each layer within the layer list, set to `true`; otherwise set to `false`. The default is `false`.
- **LayerActionsViewModel:** None
- **MapServiceActionsViewModel:** None

### See Also...

[Configure the Layer List on page 103](#)

[Menu Module on page 282](#)

## 15.39 LayerStyles Module

The LayerStyles Module enables end users to re-symbolize layers on the map. Users can create custom layer styles or use preconfigured layer styles defined in Essentials Manager.

The LayerStyles Module implements the `visualizationProvider` for layer styles. For more information see [Visualization Module on page 390](#).

Symbolization can only be modified for feature layers or layers that belong to a dynamic layers-enabled map service. Symbology must be enabled in Essentials Manager. Feature layers have symbology enabled by default. For more information about configuring symbolization, see "Symbolization" in the *Geocortex Essentials Administrator Guide*.

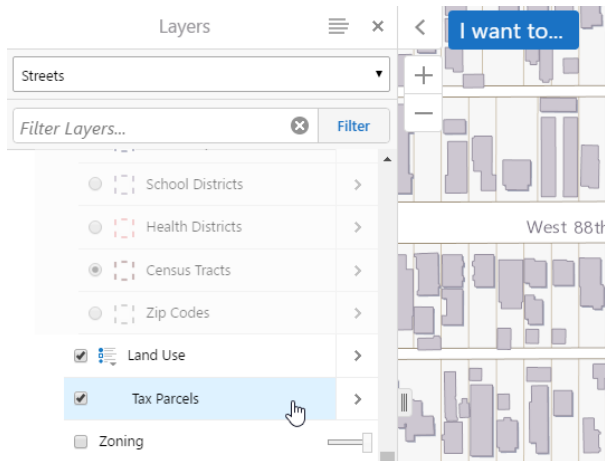
### Configure Layer Styles in the Viewer

Users can configure layer styles in the viewer from the Visualization Options panel. They can enable either a preconfigured layer style defined in Essentials Manager or define a custom layer style.



For other visualization options, see [ClusterLayers Module on page 148](#) and [HeatMaps Module on page 189](#). These modules also enable the ability to visualize layers as heat maps or cluster using the Visualization Options panel.

For example, assume the default layer style for a Tax Parcel layer is defined as shown in the following image, where the tax parcels are within the rectangular areas on the map:



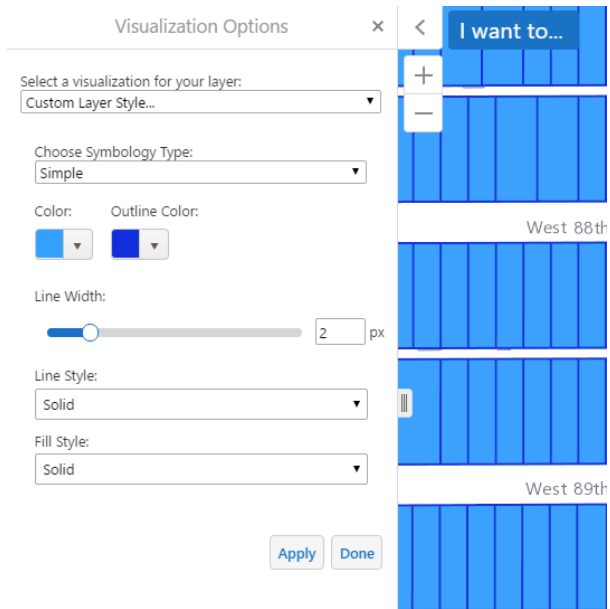
### Default layer style visualization

Re-symbolize a layer in the viewer by creating a custom layer style or by using a preconfigured layer style defined in Essentials Manager.

#### ▶ Creating a custom layer style:

1. Locate the layer (for example, Tax Parcels) in the layer list in the Viewer that you want to customize.
2. Select the ▶ Action button to the right of the layer name to display a list of layer options.
3. Click **Turn on/off layer visualizations** to open the Visualization Options panel.
4. Select **Custom Layer Style** in the drop down list.
5. Select **Simple** under **Choose Symbology Type**.
6. Choose your fill color for the tax parcels in the **Color** selector, your border color for the tax parcels in the **Outline Color** selector, then click **Apply**.  
As shown in the following image, the fill color is a solid color.

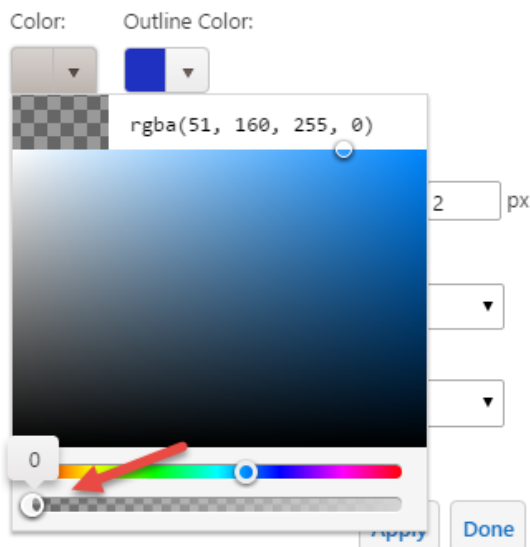




The Visualization Options panel with fill color and outline color defined for the layer

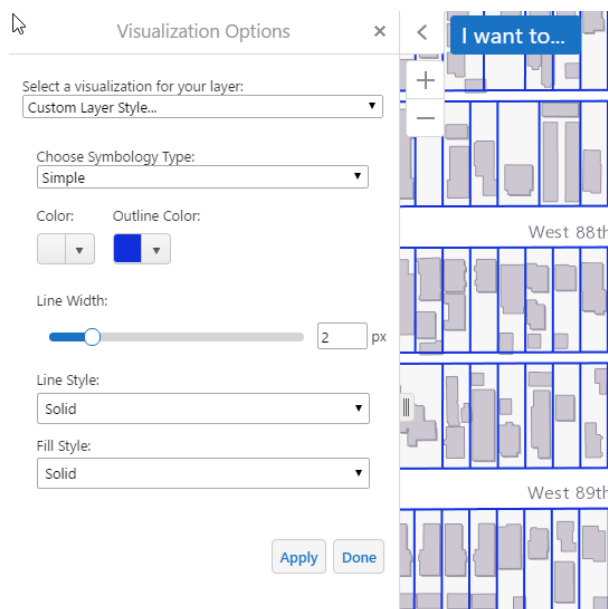
- Use the **Line Width** slider or click the numeric field and click the up/down arrows to change the pixel setting for the line width.
  - Use the **Line Style** list to choose from solid or dot and dash lines.
  - Use the **Fill Style** list to choose from a solid or patterned fill style.
7. To change the transparency/opacity setting to permit underlying layer features to show through, click the down arrow under **Color** and set the slider for the "a" (alpha) setting of the 'rgba' values to something less than 100% and click **Apply**.

In the following image, the slider is moved from 100% to 0%.



Setting the slider for the fill color to 0%

And the resulting Tax Parcels layer appears with no fill color and a dark blue outline color:



Fill color set to 0% and the outline color set to dark blue

Similar symbolization changes can be used to customize the layer style when selecting the **Attribute** symbology type .

### ► Using a preconfigured layer style defined in Essentials Manager:

1. Locate the layer in the layer list in the Viewer that you want to customize.
2. Select the ► Action button next to the layer name to display a list of layer options.
3. Select the **Turn on/off layer visualizations** to open the Visualization Options panel.
4. Select a preconfigured layer style from Essentials Manager in the drop down list.  
To learn about creating layer styles, see "Create Layer Styles that End Users can Select" in the *Essentials Administrator Guide*.  
You may also be able to visualize a layer as heat maps or clusters. See [ClusterLayers Module on page 148](#) and [HeatMaps Module on page 189](#) for more information.
5. Click **Apply**, then click **Done** or exit the Visualization Options panel.

Alternatively, you can activate the re-symbolization user interface with the `ShowVisualizationView` command. You can remove re-symbolization with the `RemoveVisualization` command.

## Default Layer Styles

The `SymbologySettings` and `AttributeSymbologySettings` widgets provide default layer style definitions that can be used for re-symbolization. You can configure the widgets from the viewer configuration files in the `widgets` section. For widget configuration documentation, see [Configuration Properties](#).

---

## Symbology Types

The widget configuration settings control two types of symbology:

- **Simple:** The Simple symbology type allows users to define the style of a single symbol for all features in a layer.
- **Attribute:** The Attribute symbology type allows users to style features in a layer based on the information in a specific field. For example, if a layer included a field with elevation data, the symbology could be rendered with up to twelve different colors, each color representing an elevation range.

## Configuration Properties

### Module

None

### Views

- **LayerStyleSelectorView:** None

### View Models

- **LayerStyleSelectorViewModel:** None

### Widgets

- **AttributeSymbologySettings:** Configures the default behavior for Attribute Symbology. This type of symbology allows users to style features in a layer based on the information in a specific field. For example, if a layer included a field with elevation data, the symbology could be rendered with up to twelve different colors, each color representing an elevation range.
  - **maxRenderClasses:** Sets the maximum number of classes that can be distinguished for the configured attribute. For example, you may wish to configure three classes for three different tree species that your layer identifies. The default value is 12. Note that increasing the default to a value greater than 12 can impact performance.
  - **maxSamples:** Defines how many features should be sampled to determine the values of a layer's render classes. The default value is 1000.
  - **defaultPointColor:** An array that defines the default color for point features in RGBA format. The default value is [150, 150, 150, 0.8].
  - **defaultPointSize:** The default size of a point feature in pixels. The default value is 12.
  - **defaultLineColor:** An array that defines the default color for line features in RGBA format. The default value is [75, 75, 75, 1].
  - **defaultLineWidth:** The default width of a line feature in pixels. The default value is 2.
  - **defaultFillColor:** The default fill color for features in RGBA format. The default value is [150, 150, 150, 0.3].
  - **defaultSymbologySettingsConfig:**

- **selectOutlineColor**: Allows the user to select an outline color. The default value is `true`.
- **alwaysUseColorSwatches**: Allows the user to always choose colors using color pickers. The default value is `false`.
- **numberOfColorSwatches**: Defines how many color pickers should be available from the attribute symbology settings. The default value is 6.
- **transparency**: Sets the user configuration options for the transparency slider. These properties accept values between 0 and 100 percent, where 0 is opaque and 100 is transparent.
  - **min**: Sets the minimum transparency value available, in percent. The default value is 0.
  - **max**: Sets the maximum transparency value available, in percent. The default value is 90.
  - **value**: Sets the initial transparency value, in percent. The default value is 10.
  - **step**: Sets the increment to use when the user increases or decreases the transparency value, in percent. For example, if the value is 10, there are 10 unique transparency values that can be picked between 0 and 100. The default value is 5.
- **lineWidth**: Sets the user configuration options for the symbol line width slider.
  - **min**: Sets the minimum configurable width of the line, in pixels. The default value is 0.
  - **max**: Sets the maximum configurable width of the line, in pixels. The default value is 5.
  - **value**: Sets the initial configured line width value, in pixels. The default value is 2.
  - **step**: Sets the increment to use when the user increases or decreases the line width, in pixels. The default value is 1.
- **markerSize**: Sets the user configuration options for the marker size slider.
  - **min**: Sets the minimum configurable marker size, in pixels. The default value is 1.
  - **max**: Sets the maximum configurable marker size, in pixels. The default value is 50.
  - **value**: Sets the initial configured marker size value, in pixels. The default value is 16.
  - **step**: Sets the increment to use when the user increases or decreases the marker size, in pixels. The default value is 1.
- **markerStyles**: An array of objects that define available marker styles. By default, the styles included are `circle`, `diamond`, `cross`, `x`, and `square`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style**: A string that references an available marker style.
  - **label**: The label to display for the marker style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for marker style language keys are `@language-symbology-settings-marker-style-*`, where `*` is the marker style —`circle`, `diamond`, and so on.
- **lineStyles**: An array of objects that define available line styles. By default, the styles included are `solid`, `dash`, `dot`, and `dashdot`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).

- **style**: A string that references an available line style.
- **label**: The label to display for the line style in the viewer. The line style is the style of the border of the feature marker. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for line style language keys are `@language-symbology-settings-line-style-*`, where `*` is the marker style —`solid`, `dash`, and so on.
- **fillStyles**: An array of objects that define available fill styles. The fill is the most dominant pattern of the feature marker. By default, the styles included are `solid`, `forwarddiagonal`, `backwarddiagonal`, `cross`, `horizontal`, and `vertical`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style**: A string that references an available fill style.
  - **label**: The label to display for the fill style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for fill style language keys are `@language-symbology-settings-fill-style-*`, where `*` is the marker style —`solid`, `forward-diagonal`, `backward-diagonal`, and so on.
- **SymbologySettings**: Configures the default behavior for Simple Symbology.
  - **alwaysUseColorSwatches**: Defines whether color pickers should always be available. The default value is `false`.
  - **numberOfColorSwatches**: Defines how many color pickers should be available from the attribute symbology settings. The default value is `6`.
  - **transparency**: Sets the user configuration options for the transparency slider. These properties accept values between `0` and `100` percent, where `0` is opaque and `100` is transparent.
    - **min**: Sets the minimum transparency value available, in percent. The default value is `0`.
    - **max**: Sets the maximum transparency value available, in percent. The default value is `90`.
    - **value**: Sets the initial transparency value, in percent. The default value is `10`.
    - **step**: Sets the increment to use when the user increases or decreases the transparency value, in percent. For example, if the value is `10`, there are `10` unique transparency values that can be picked between `0` and `100`. The default value is `5`.
  - **lineWidth**: Sets the user configuration options for the symbol line width slider.
    - **min**: Sets the minimum configurable width of the line, in pixels. The default value is `0`.
    - **max**: Sets the maximum configurable width of the line, in pixels. The default value is `5`.
    - **value**: Sets the initial configured line width value, in pixels. The default value is `2`.
    - **step**: Sets the increment to use when the user increases or decreases the line width, in pixels. The default value is `1`.

- **markerSize:** Sets the user configuration options for the marker size slider.
  - **min:** Sets the minimum configurable marker size, in pixels. The default value is 1.
  - **max:** Sets the maximum configurable marker size, in pixels. The default value is 50.
  - **value:** Sets the initial configured marker size value, in pixels. The default value is 16.
  - **step:** Sets the increment to use when the user increases or decreases the marker size, in pixels. The default value is 1.
- **markerStyles:** An array of objects that define available marker styles. By default, the styles included are `circle`, `diamond`, `cross`, `x`, and `square`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style:** A string that references an available marker style.
  - **label:** The label to display for the marker style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for marker style language keys are `@language-symbology-settings-marker-style-*`, where `*` is the marker style —`circle`, `diamond`, and so on.
- **lineStyles:** An array of objects that define available line styles. By default, the styles included are `solid`, `dash`, `dot`, and `dashdot`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style:** A string that references an available line style.
  - **label:** The label to display for the line style in the viewer. The line style is the style of the border of the feature marker. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for line style language keys are `@language-symbology-settings-line-style-*`, where `*` is the marker style —`solid`, `dash`, and so on.
- **fillStyles:** An array of objects that define available fill styles. The fill is the most dominant pattern of the feature marker. By default, the styles included are `solid`, `forwarddiagonal`, `backwarddiagonal`, `cross`, `horizontal`, and `vertical`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style:** A string that references an available fill style.
  - **label:** The label to display for the fill style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for fill style language keys are `@language-symbology-settings-fill-style-*`, where `*` is the marker style —`solid`, `forward-diagonal`, `backward-diagonal`, and so on.

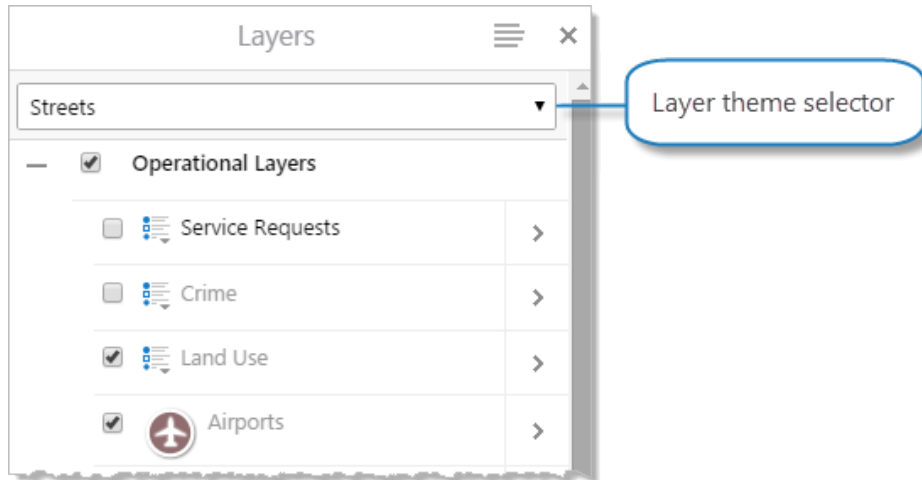
## 15.40 LayerThemes Module

The LayerThemes Module implements the ability to switch between the layer themes of the map. However, it is the [LayerList Module](#) that actually provides the layer theme selector in the layer list.

You can use Manager to configure which layer themes are available for the site. For more information, see *Layer Themes* in the *Geocortex Essentials Administrator Guide*.



It is possible to launch the viewer with a specific layer theme by using the `layerTheme` URL parameter, along with the layer theme ID or Display Name. For more information, see [URL Parameters Reference on page 49](#).



Layer List with the Streets layer theme selected

## Configuration Properties

Module

None

Views

None

View Models

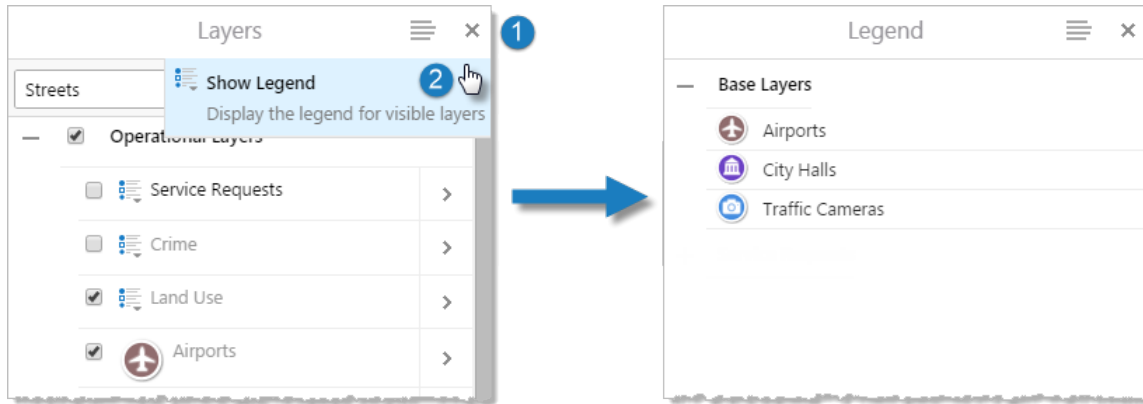
None

**See Also...**

[LayerList Module on page 232](#)

## 15.41 Legend Module

The Legend Module implements the map legend. To show the Legend panel, open the **Layers** panel, open the **Panel Actions Menu** <sup>1</sup> and select **Show Legend** <sup>2</sup>.



### Open the legend

Users can also view legend switches in the Layers panel. See [LayerList Module on page 232](#) for information.

## Configuration Properties

### Module

None

### Views

- **LegendView:** None

### View Models

- **LegendViewModel:** None



## 15.42 Log Module

The Log Module tracks and logs tasks and events for debugging.

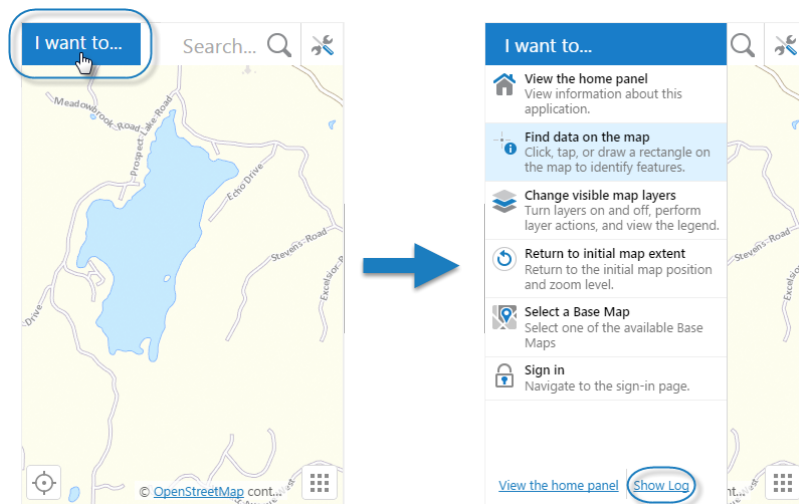
### ► To view the log file:

- **On a desktop PC or tablet:** Press **Ctrl + Shift + ~**.  
Alternatively, hold down the **Shift** key and press **Esc** twice.

Debug Log		
Level	Timestamp	Message
info	15:06:25.746	Version 1.1.0.0751. en-us
info	15:06:25.746	Initializing module 'Alert'.
info	15:06:25.746	Initializing module 'Authentication'.
info	15:06:25.746	Initializing module 'Banner'.
info	15:06:25.746	Initializing module 'Confirm'.
info	15:06:25.747	Initializing module 'FeatureDetails'.
info	15:06:25.747	Initializing module 'FeatureLayer'.
info	15:06:25.748	Initializing module 'Identify'.
info	15:06:25.749	Initializing module 'IWantToMenu'.
info	15:06:25.750	Initializing module 'Map'.
info	15:06:25.751	Initializing module 'MapTips'.
info	15:06:25.751	Initializing module 'Menu'.
info	15:06:25.751	Initializing module 'Offline'.
info	15:06:25.752	Initializing module 'Prompt'.

Example of a debug log

- **On a handheld device:** Tap the **I Want To** menu button, and tap **Show Log**.



The I Want To menu button in the Handheld interface (left), and the Show Log link

## Configuration Properties

### Module

None

### Views

- **LogView:** None

### View Models

- **LogViewModel1:** None

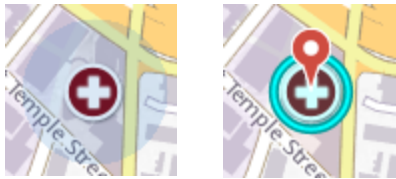
## 15.43 Map Module

The Map Module manages the map and map-related events.

### Visual Feedback when Clicking a Map

A built-in default provides a visual cue to indicate where the user clicked in the map when no other tool is active.

The visual cue is a faint ripple effect, which is followed immediately by the appropriate map action for the clicked target. For example, when a user clicks a map feature, the ripple effect occurs, a pushpin appears on the target, the target in focus is highlighted in blue, and a map tip opens.



Ripple effect on a map (left image) followed by a pushpin and focus highlighting on the target (right image)

## Configuration Properties

### Module

- **panDuration:** The length of time in milliseconds that the map takes to pan from one extent to another. The default is 350 milliseconds.
- **panRate:** The length of time in milliseconds that the map takes to refresh as it pans to the next extent. The default is 50 milliseconds.
- **zoomDuration:** The length of time in milliseconds that the map takes to zoom from one extent to another. The default is 500 milliseconds.
- **zoomRate:** The length of time in milliseconds that the map takes to refresh as it zooms to the next extent. The default is 50 milliseconds.



`panDuration`, `panRate`, `zoomDuration`, and `zoomRate` are Esri map control properties.

- **longClickMilliseconds:** The number of milliseconds the user must hold down the mouse button or finger on the touchscreen, before the commands specified in `onLongClick` execute. The default is 500.
- **maxExtentLogSize:** The maximum number of map extent changes available when zooming to the previous or next extent. The default for the Desktop and Tablet interface is 100. The default for the Handheld interface is 50.
- **showLoadingStatus:** When set to `true`, shows a **Loading** status message while loading the map. The default is `true`.
- **loadingMessageTiming:** An array of times in milliseconds to elapse before a stage 1, stage 2, and stage 3 "Still loading" messages display. The defaults are 1000 ms, 3000 ms, and 3000 ms respectively.
- **defaultPointFeatureZoomScales:** An array of default zoom scales for point features. The default is an empty array.
- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **MapOnClickBehavior:** A behavior that runs an array of commands when the user quickly clicks on or touches the map. By default, this includes the command: `InvokeMapTip`.
  - **MapOnContextMenuBehavior:** A behavior that runs an array of commands when the user right-clicks on the map. By default, this includes the command: `ActivateMapContextMenu`.
  - **MapOnLongClickBehavior:** A behavior that runs an array of commands when the user clicks on or touches the map for the amount of time specified by `longClickMilliseconds`. By default, this includes the command: `ActivateMapContextMenu`.
  - **MapOnFeatureClickBehavior:** A behavior that runs an array of commands, and works in conjunction with the MapTips module's `webMapFeaturePresenter` property to show map tips for features from web maps. If the site does not contain any references to web maps, this property has no effect. By default, this includes the command: `ShowMapTip`.



Modifying the `MapOnFeatureClickBehavior` property is an advanced task. We recommend that you use the default configuration for `MapOnFeatureClickBehavior`.

- **MapTimeExtentChangedBehavior:** A behavior that runs an array of commands when the map's time extent is changed. The default is an empty array.



You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.



You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.



Adding a new behavior is only recommended for experienced developers.

- **menus:** An array of menus used by the Map Module. By default, this includes a single menu: `MapContextMenu`. Each menu has the following properties:
  - **id:** The menu's ID.
  - **defaultIconUri:** The URI of the default icon for menu items.
  - **items:** An array of menu items. Menu items have the following properties:
    - **iconUri:** The image to display beside the menu item. The recommended image size is 24px by 24px.
    - **text:** The text to appear on the menu item. You can use a text key or the literal text.
    - **description:** A brief description of the menu item to appear below the `text`. You can use a text key or the literal text.
    - **command:** The command to execute when the menu item is selected.



If you set the `command` property, do not set the `batch` property.

- **commandParameter:** The parameter value to pass to the command when it runs, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.



If you set the `commandParameter` property, do not set the `batch` property.

In the following example from the `MapContextMenu`, a `commandParameter` is passed into the `PanToPoint` command to center the map on the point the user right-clicked or long-pressed:

```
"command": "PanToPoint",
"commandParameter": "{{context}}"
```

The `{{context}}` token is a special token that retrieves the default menu context which a menu may have. For example, the default context of `MapContextMenu` is the `Point` the user right-clicked or long-pressed on the map. Therefore, the command centers the map to the point that the user right-clicked or long-pressed.

- **hideOnDisable:** If this property is set to `true` and the menu item's command cannot run, the menu item does not show in the menu.

If `hideOnDisable` is `false` and the menu item's command cannot run, the menu item shows in the menu, but it is grayed out.

- **batch**: An array of objects, each with three properties: an executable command, an optional command parameter, and an optional Boolean to abort the execution of further commands if a command fails to execute. This allows multiple commands to be executed in a specified order, although some commands may trigger asynchronous actions.

The following example attempts to activate the Tabbed Toolbar and, if successful, proceeds to greet the user with an alert:

```
"batch": [
  {
    "command": "ActivateView",
    "commandParameter": "TabbedToolbarView",
    "abortBatchOnFailure": true
  },
  {
    "command": "Alert",
    "commandParameter": "Hello!"
  }
]
```

If omitted, `abortBatchOnFailure` is `false` by default.



If you set the `batch` property, do not set the `command` or `commandParameter` properties.

- **tools**: An array of tools related to working with the map. The factory configuration has tools to center the map, zoom in, and zoom out.

The Tools Module implements the ability to define an array of tools in other modules. Each tool in the array has the following properties:

- `name`: The name of the tool.



If your viewer is going to be available in more than one language, enter the text key that the tool name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- `command`: The command that the tool runs.  
For a list of commands, see the [Geocortex SDK for HTML5 API Reference](#).
- `drawMode`: The type of geometry the user draws, upon which the tool operates.
- `isSticky`: When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.

If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.

- `iconUri`: The image that displays beside the tool.
- `statusText`: The status message to display when the tool's input method is via mouse, often containing instructions for the user. You can use a text key or the literal text.
- `keyboardStatusText`: The status message to display when the tool's input method is via keyboard, often containing keyboard shortcut hints for the user. You can use a text key or the literal text.

## Views

- **MapView:**

- `wrapAround180`: When set to `true`, the map supports continuous pan across the date line. The default is `false`.
- `disableMapKeyboardNavigation`: Disables keyboard navigation for the map when set to `true`. The default value is `false`.



When you choose to disable map keyboard navigation, note that your viewer is no longer Web Content Accessibility Guidelines 2.0 compliant. Users who require the keyboard to navigate on the map may be unable to use your viewer. For additional information, see [Accessibility on page 402](#).

- `extentBroadcastFrequency`: The frequency in milliseconds after which the `MapExtentChanging` event is published. The default is 20 ms.
- `fitTiledMapsToExtent`: If set to `true`, a map containing tiled map services is guaranteed to display the entirety of its initial extent. The default is `false`.
- `showAttribution`: If set to `true`, copyright information is displayed about the map if available. The default is `true`.
- `minScale`: The minimum scale at which the map is visible in viewers. For example, if the map's minimum scale is 1:5,000,000, you could extend the minimum scale to 1:10,000,000 in Essentials by setting `minScale` to **10000000**. You can also configure this setting in Manager, on the viewer's Map page.
- `maxScale`: The maximum scale at which the map is visible in viewers. For example, if the map's maximum scale is 1:2,000, you could extend the maximum scale to 1:500 in Essentials by setting `maxScale` to **500**. You can also configure this setting in Manager, on the viewer's Map page.

- **MapContextMenuView: None**





The `MapContextMenuView` has a `title` property that determines the panel title used for the map context menu in the Handheld interface.

- **ContextMenuReverseGeocodeView: None**
- **ContextMenuCoordinatesView: None**

- **ContextMenuMapView:**
  - `menuId`: The ID of the menu to use in the map context menu. The default is `MapContextMenu`.

## View Models

- **MapViewModel:**
  - `stepZoomFactor`: The factor by which the map zooms in or out when the user clicks the Zoom In map widget  or the Zoom Out map widget . The Zoom In and Zoom Out map widgets are implemented by the [ZoomControl Module](#).  
The value that you specify for `stepZoomFactor` is inversely proportional to the size of the step. To decrease the amount that the extent changes each time the user clicks the widget, specify a larger value. To change the extent more with each step, specify a smaller value.  
The value must be between 0 and 1. The default value is 0.5.



If you are using a tiled map service, make sure the step size is large enough to reach the next cached map scale with each widget click—the HTML5 Viewer cannot show extents between two cached scales in a tiled map service.

- **MapContextMenuViewModel: None**
- **ContextMenuReverseGeocodeViewModel:**
  - `showReverseGeocoder`: To display the address of the point right-clicked or long-pressed, set to `true`; otherwise, set to `false`. The default value is `true`.
- **ContextMenuCoordinatesViewModel:**
  - `showCoordinates`: To display the coordinates of the point right-clicked or long-pressed, set to `true`; otherwise, set to `false`. The default value is `true`.
- **ContextMenuMapViewModel:**
  - `showMenu`: To display the menu items of the Map Context Menu, set to `true`; otherwise, set to `false`. The default value is `true`.
- **MapCoordinatesModel:**
  - `defaultCoordinateDisplayTypes`: An array of display formats to present coordinates. By default, the following are included:
    - `dd`: Decimal degrees.
    - `ddm`: Decimal degrees with minutes.
    - `dms`: Degrees, minutes, seconds.
    - `xy`: Native X and Y coordinates.
  - `customCoordinateSystems`: An array of custom coordinate systems. By default, the array is empty.
  - `fractionalDigits`: An integer that specifies the number of decimal places to display for coordinates.

The default value is 5.

- `defaultGcsWkid`: The default Geographic Coordinate System to display. The default value is 4326.

## Example: Add a Command with a Parameter to a Behavior

The following example demonstrates adding a new command with a parameter to the behavior, `MapOnFeatureClickBehavior`.

### To add a command with a parameter to a behavior:

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. In the `Map` module section, find the `behaviors` property. Locate the behavior you want to edit. For example, `MapOnFeatureClickBehavior`.

```
{
  "moduleName": "Map",
  ...
  "configuration": {
    ...
    "behaviors": [
      ...
      {
        "name": "MapOnFeatureClickBehavior",
        "commands": [
          "ShowMapTip"
        ]
      }
    ]
  },
  ...
}
```

The behavior executes a single command: `ShowMapTip`.

4. Consult the Geocortex SDK for HTML5 API Reference to determine the type of parameter that is associated with this command. `ShowMapTip` has a parameter of type, `geocortex.essentialsHtmlViewer.mapping.infrastructure.Feature`.
5. Find a command that you want to add to the behavior in the Geocortex SDK for HTML5 API Reference that has



the same parameter type. For example, `ZoomToFeature`.

6. Add the desired command to the list of commands, separated by a comma. For example:

```
"commands": [
  "ShowMapTip",
  "ZoomToFeature"
]
```

7. Save the file.

#### See Also...

[Geocortex SDK for HTML5 API Reference on page 422](#)

[Tools Module on page 375](#)

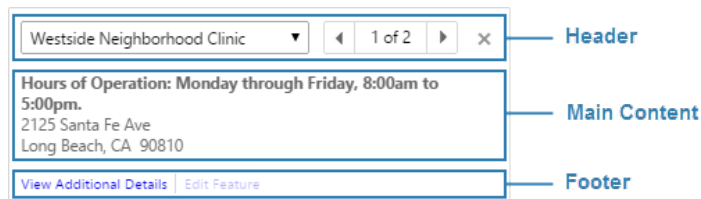
[About User Interface Text on page 64](#)

[ZoomControl Module on page 395](#)

## 15.44 MapTips Module

The MapTips module implements map tips in the HTML5 Viewer. Map tips are the pop-up windows that appear on the map when the user clicks on the map.

In order for map tips to work in an HTML5 viewer, you must enable map tips for each layer that you want to support map tips. You must also configure the content that you want to appear in map tips. The header and main content are configured in the layer. See "Configure Map Tips" in the *Geocortex Essentials Administrator Guide* for information.



#### Example of a map tip, showing the configurable parts

The footer content is configured in the Menu Module's `MapTipActions` menu. See [Menu Module on page 282](#).

You can also customize the message that displays in map tips when no results are returned. Use the Results module's `customSearchSuggestions` property to customize the message. See [Results Module on page 328](#) for information.

The Identify module performs the query when a user clicks the map. See [Identify Module on page 193](#) for information.

Before version 2.4 of the HTML5 Viewer, map tips displayed in callouts. Starting in version 2.4, the HTML5 Viewer supports both fixed-location and callout-style map tips.

### Fixed-Location Map Tips

By default, HTML5 viewers display map tips at a fixed location. For example, the Desktop and Tablet interfaces show map tips in the `NavigationMapRegion` by default.

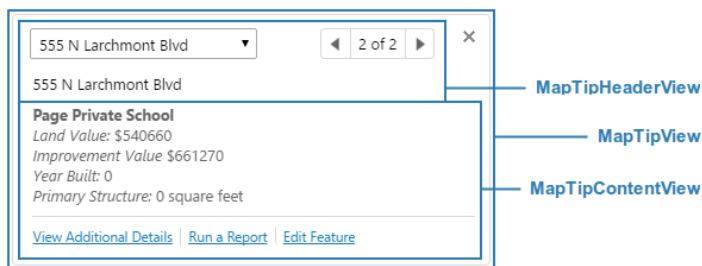
The HTML5 Viewer has the following commands for working with fixed-location map tips:

- `InvokeMapTip` (works with both fixed-location and callout-style map tips)
- `ShowMapTip`
- `ShowMapTipResults`
- `AddPushpin`
- `RemovePushpin`

For more information about these commands, see "Commands" in the [Geocortex SDK for HTML5 API Reference](#).

The MapTips module has three views—`MapTipHeaderView`, `MapTipHeaderView`, and `MapTipContentView`. Fixed-location map tips use all three of these views.

`MapTipHeaderView` contains the map tip's header, controls, and title. `MapTipContentView` contains the map tip's main content and footer links. Both of these views appear within `MapTipView` in the viewer. The three views all share the same view model, `MapTipViewModel`.



### Views used by fixed-location map tips

## Callout-Style Map Tips

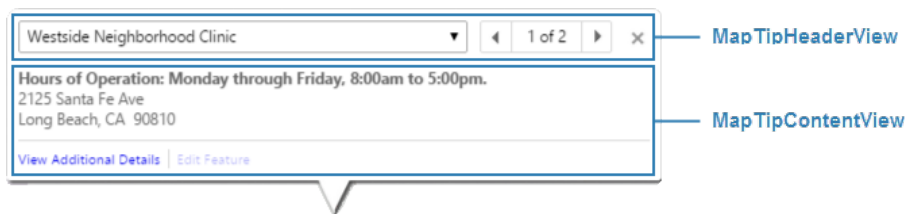
You can configure an HTML5 viewer to use callout-style map tips instead of fixed-location map tips. To do this, you replace the fixed-location version of map tip commands with the callout-style version of the commands in the configuration files. The HTML5 Viewer has the following commands for working with callout-style map tips:

- `InvokeMapTip` (works with both fixed-location and callout-style map tips)
- `ShowMapTipInCallout`
- `ShowMapTipResultsInCallout`

For more information about these commands, see "Commands" in the [Geocortex SDK for HTML5 API Reference](#).

Callout-style map tips use two of the MapTips module's three views—`MapTipHeaderView` and `MapTipContentView`. `MapTipHeaderView` contains the map tip's header. `MapTipContentView` contains the map tip's main content and footer. The callout itself is built into the HTML5 Viewer—the callout does not have a view.

The views share the same view model, `MapTipViewModel`.



### Views used by callout-style map tips

### ► To configure an HTML5 viewer to use callout-style map tips:

1. In a text editor or XML editor, open one of the viewer configuration files—`Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`.  
By default, the configuration files are stored in the Sites folder:  
`Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default`
2. Replace each instance of `ShowMapTipResults` with `ShowMapTipResultsInCallout`.
3. Replace each instance of `ShowMapTip` with `ShowMapTipInCallout`.
4. Save the file.
5. Repeat these steps for each configuration file.

#### Map Tips with Null Geometry

When a feature is selected in the Results Table and it has no spatial data, a status message notifies the user with the following message: **"The selected feature contains no spatial data and cannot be highlighted on the map."**

If a selected feature does not have spatial data, the map tip displays a message, **"No spatial data"** in the map tip content.

## Configuration Properties

### Module

- **allowMultiple**: When set to `true`, multiple callout-style map tips can display simultaneously on the map. When set to `false`, only one callout-style map tip can display at a time. The default is `false`. This property has no effect on fixed-location map tips.
- **contentField**: Specifies whether to display the Feature Description or Feature Long Description in map tips. The default is `longDescription`. To display the content defined in the Feature Description instead, set `contentField` to `description`.



You can also configure the content field in Manager, on the viewer's Look and Feel page.

- **behaviors**: An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **MapTipOnCloseBehavior**: A behavior that runs an array of commands when the user closes a fixed-location map tip. By default, this includes the command: `StopEditingClickableFeature`.
  - **MapCalloutClosedBehavior**: A behavior that runs an array of commands when the user closes a callout-style map tip. The default is an empty array.
  - **MapTipFeatureChangedBehavior**: A behavior that runs an array of commands when a feature is first presented in a fixed-location map tip, or when the user selects a different feature in a fixed-location map tip. This property has no effect on callout-style map tips. By default, this includes the commands: `PanToFeatureIfOutsideMapExtent` and `StopAndAutoEditClickableFeature`.

- **MapCalloutFeatureChangedBehavior**: A behavior that runs an array of commands when a feature is first presented in a callout-style map tip, or when the user selects a different feature in a callout-style map tip. This property has no effect on fixed-location map tips. By default, this property is omitted.



You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.



You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.



Adding a new behavior is only recommended for experienced developers.

- **webMapFeaturePresenter**: Works in conjunction with the Map module's `onFeatureClick` property to show map tips for features from web maps. If the site does not contain any references to web maps, the `webMapFeaturePresenter` property has no effect.



Modifying the `webMapFeaturePresenter` property is an advanced task. We recommend that you use the default configuration for the `webMapFeaturePresenter`.

- **nullGeometryStatusMessageEnabled**: When set to `true`, a status message appears if a selected feature has null geometry. The default value is `true`.
- **nullGeometryStatusMessageArgs**: If `nullGeometryStatusMessageEnabled` is set to `true`, a status message requires the following properties:
  - **imageUri**: The URI of the alert icon displayed with the status message. The default value is `@language-map-tip-null-geometry-status-message-uri`.
  - **timeoutMS**: The length of time that the status message displays in milliseconds. The default value is `5000`.

## Views

- **MapTipView**: None
- **MapTipHeaderView**: None
- **MapTipContentView**: None

## View Models

- **MapTipViewModel**: None

## Example: Add a Command with a Parameter to a Behavior

The following example demonstrates adding a new command with a parameter to the behavior, `MapOnFeatureClickBehavior`.

### ► To add a command with a parameter to a behavior:

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Sites\[site]\Viewers\
[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. In the `MapTips` module section, find the `behaviors` property. Locate the behavior you want to edit. For example, `MapTipFeatureChangedBehavior`.

```
{
  "moduleName": "MapTips",
  ...
  "configuration": {
    ...
    "behaviors": [
      ...
      {
        "name": "MapTipFeatureChangedBehavior",
        "commands": [
          "PanToFeatureIfOutsideMapExtent"
        ]
      }
    ]
  },
  ...
}
```

The behavior executes a single command: `PanToFeatureIfOutsideMapExtent`.

4. Consult the Geocortex SDK for HTML5 API Reference to determine the type of parameter that is associated with this command. `PanToFeatureIfOutsideMapExtent` has a parameter of type, `geocortex.essentialsHtmlViewer.mapping.infrastructure.Feature`.
5. Find a command that you want to add to the behavior in the Geocortex SDK for HTML5 API Reference that has the same parameter type. For example, `PanToFeature`.
6. Add the desired command to the list of commands, separated by a comma. For example:

```
"commands": [  
  "PanToFeatureIfOutsideMapExtent",  
  "PanToFeature"  
]
```

7. Save the file.

**See also...**

[Geocortex SDK for HTML5 API Reference on page 422](#)

[Identify Module on page 193](#)

[Menu Module on page 282](#)

[Results Module on page 328](#)

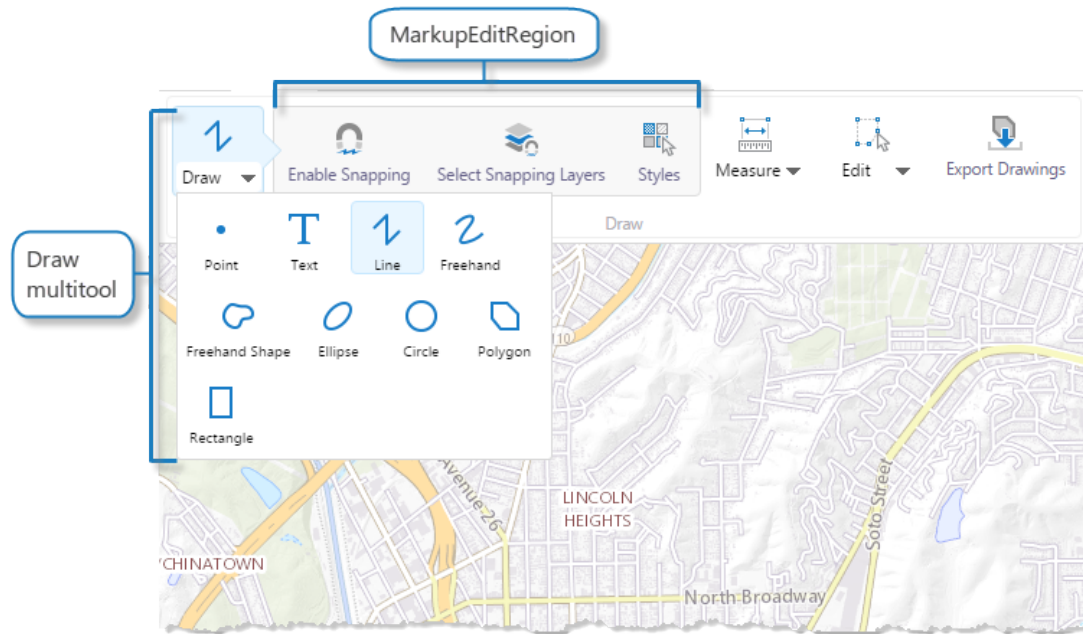
## 15.45 Markup Module

The Markup Module implements the ability for end users to add markup text and shapes to the map. Markup is added to the map's drawing layer. The drawing layer does not appear in the layer list.

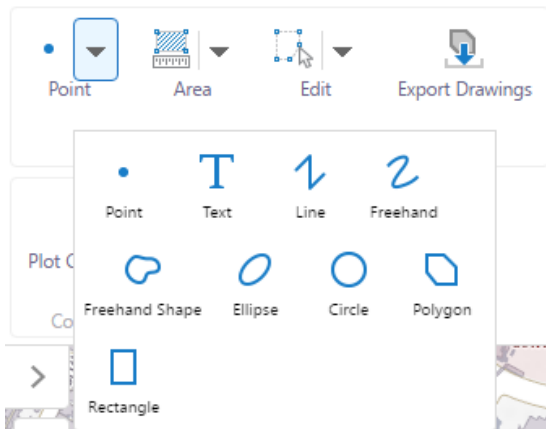
Markup exists only for the duration of the session—when the user closes the viewer, the markup disappears. To save markup across sessions and share it with other users, a signed-in user can save a project, provided the viewer supports projects. For information about projects, see [Project Module on page 304](#).




The Markup Module is integrated with the [Measurement Module](#). For more information on how measurements affect this module, see the section [Markup Measurement](#).



Drawing creation tools in the Desktop and Tablet interfaces



Drawing creation tools in the Handheld interfaces

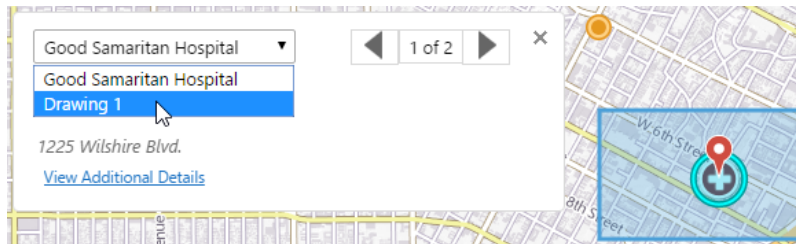
 **Text markup limitation:** The styling options for the halo (outline) effect on text markup – color and width – are not available if the HTML5 Viewer is used with Internet Explorer versions 9 or 11. Text halo styling options are available in all other supported browsers. See [Client Requirements on page 4](#) for information about supported browsers and devices.

## Edit Drawings

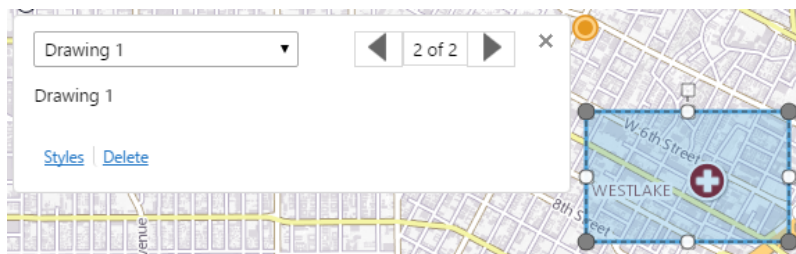
The capability to include and provide editing of drawings on a map tip is dependent on "enabled" set to true on the `GraphicsLayerIdentifyProvider` in the configuration of the Identify module. If "enabled" is set to false, the include and edit capability on a map tip is not available.

When a user adds a drawing to the map:

- If they click a feature overlaid by the drawing, the feature and the drawing are included in the map tip.  
Depending on the zoom level, a pushpin identifies the feature's location on the map.
  - To edit the drawing included in the map tip, open the drop down list and select the drawing name to place it in edit mode. (Alternatively, click the next page or previous page arrows to go to the drawing page in the map tip.)



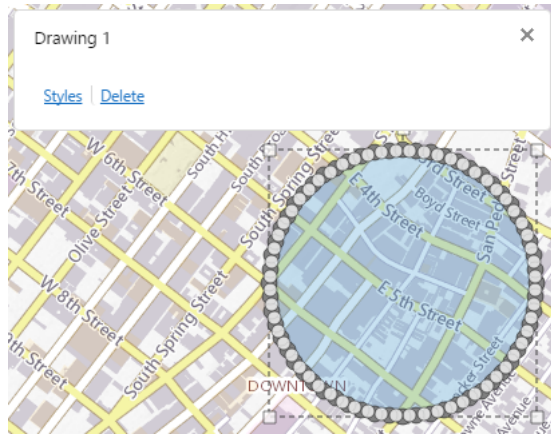
Select Drawing 1 to edit



Drawing 1 selected and in edit mode

- If they click a drawing that does not include another feature:
  - A map tip opens identifying the drawing.
  - The drawing is placed in edit mode.
  - Edit handles (not applicable to Points) are visible.



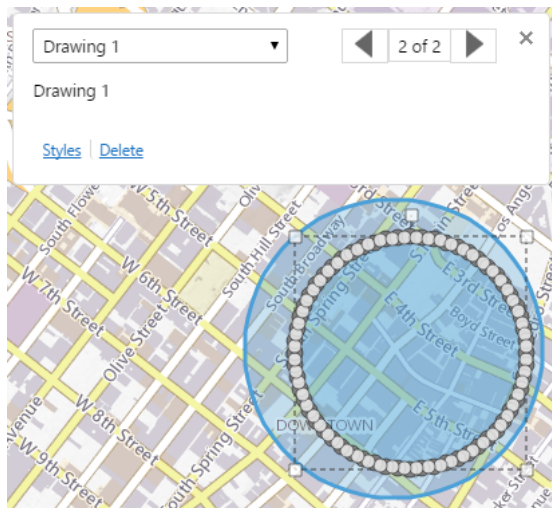


Map tip for Drawing 1 and its appearance in edit mode

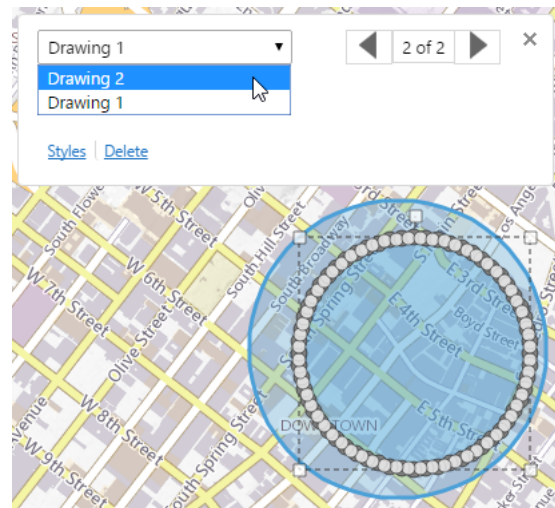
When the drawing is in edit mode:

- Depending on the type of drawing (point, text, circle, rectangle and so on), the user can edit text, add, delete or move vertices, relocate the drawing on the map, rotate it, and resize it.
- The user can change the styles for the drawing by clicking **Styles** in the map tip to open the **Select Drawing Style** panel. See [Markup Styles on page 264](#) for more information.
- The user can edit drawings that are overlaid on the map.

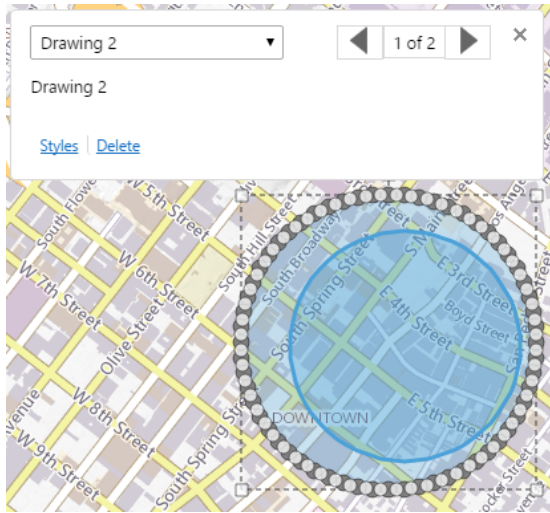
Each drawing can be selected in the map tip, either from the drop down list or by using the next page and previous page arrows.



Drawing 1 in edit mode



Select Drawing 2 to edit



Drawing 2 selected and in edit mode

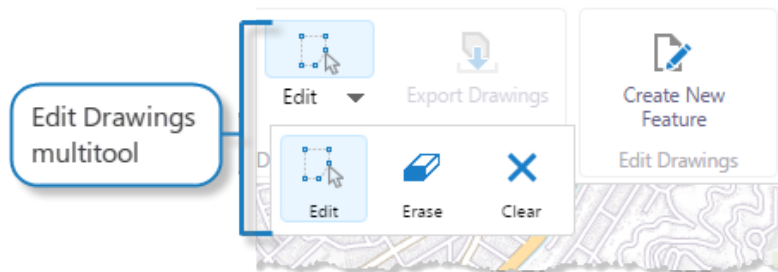
The user can also click a measurement drawing or a collaboration graphic to place it in edit mode. See [Measurement Module on page 269](#) and [Collaboration Module on page 149](#) for further information.

Note that drawings, measurement drawings, and collaboration graphics are included in the results of an identify operation.

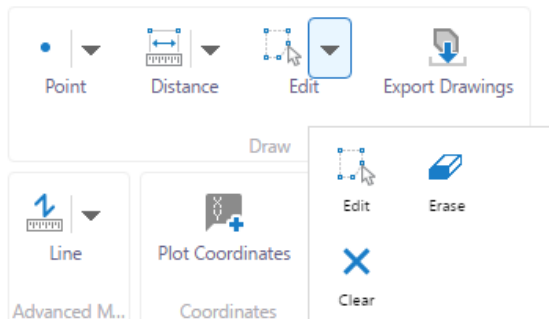
## Editing Tools

Clicking a drawing to access the editing function via the map tip is preferable to using the **Edit Drawings** multitool in the toolbar.

**Edit Drawings** in the toolbar allows the user to edit, erase, or clear drawings. Note that using **Erase** on drawings that overlay each other erases all of the drawings. To avoid this behavior, the user can click one of the drawings to open the map tip, select the drawing to be deleted from the drop down list, and click **Delete**.



Edit tools for drawings in the Desktop and Tablet interfaces



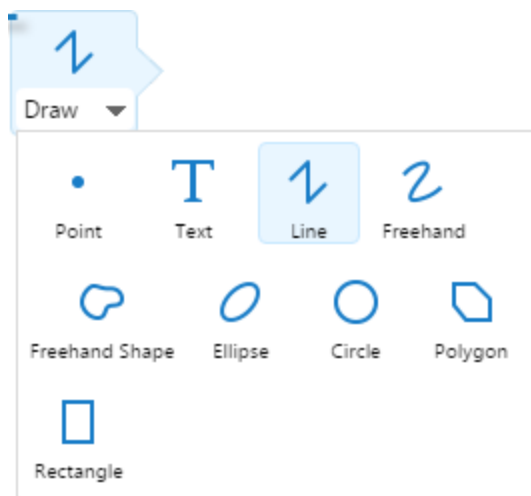
### Edit tools for drawings in the Handheld interfaces

## Configure the Toolbar

The Markup Module provides tools for working with markup. These tools are not in the default toolbar—you must add them to the toolbar. If you use the Web GIS (Full) Toolbar, the markup tools are available in the Draw section on the Tools tab. For instructions on adding tools to the toolbar, see [Configure the Toolbar on page 108](#).

The Markup Module makes the following tools available:

- **Draw (multitool):** Drawing tools for various markup types.



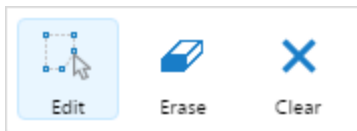
When the Draw multitool is selected, various markup tools become available.

- **MarkupEditRegion:** Select [snapping](#) rules and styles for new markup. For more information about the Styles tool, see [Markup Styles](#).



The MarkupEditRegion tools, Enable Snapping, Select Snapping Layers, and Styles.

- **Edit Drawings (multitool):** Tools to edit, erase, and clear drawings. If the Edit tool is selected, the Erase and Clear tools can be activated. These tools apply to both drawings and measurements.



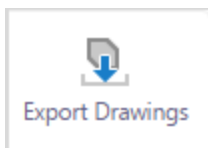
With the **Edit** tool activated, the **Erase**, and **Clear** tools become available.

- **EditControlRegion:** Edit [snapping](#) rules and styles for existing markup. For more information about the **Styles** tool, see [Markup Styles](#).




The **MarkupEditRegion** tools, **Enable Snapping**, **Select Snapping Layers**, and **Styles**.

- **Export Drawings:** Export all drawings from the map to shapefiles. For more information, see [Export Drawings](#).



The **Export Drawings** tool.

The **Export Drawings** feature can also be configured to the **I Want To** menu with the `ExportMarkupLayer` command.

 The **Drawing** and **Edit Drawing** groups are included on the **Web GIS (Full) Toolbar** preset.

## Markup Styles

When a user clicks a drawing to place it in edit mode, they can click **Styles** in the map tip to open the **Select Drawing Style** panel. Each type of drawing has a specific set of style options to choose from.

For example, to change from the default point style to an enlarged, angled image point style:

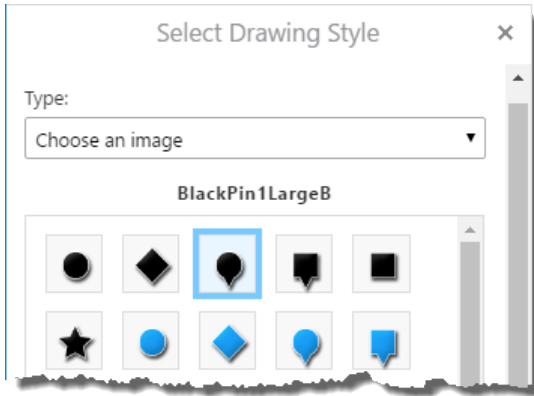


Default point style on a map

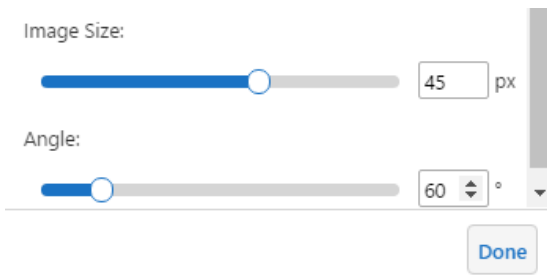


Image point style on a map

1. Click **Styles** in the map tip to open the **Select Drawing Style** panel.
2. Select **Choose an image** in the **Type** drop down list.

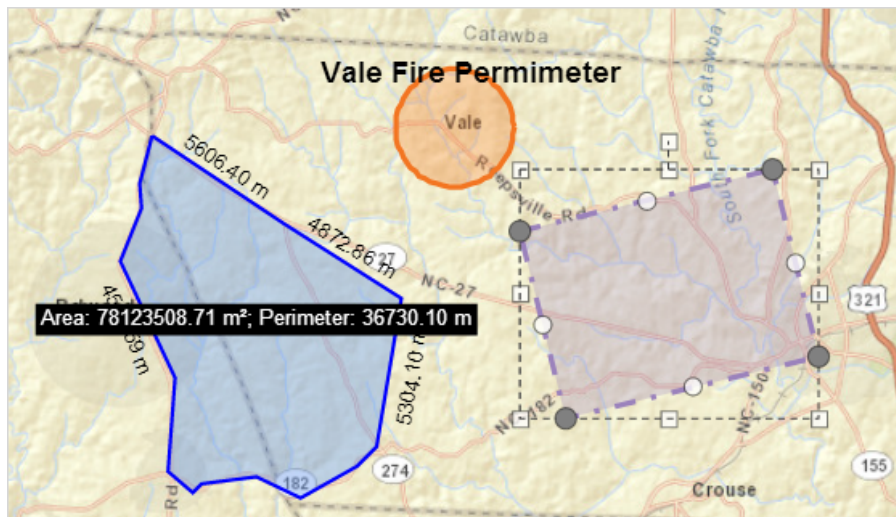


3. Click a black pin image.
4. Change **Image Size** from 32 px to 45 px, and **Angle** from 0 degrees to 60 degrees.



5. Click **Done** to apply the style changes to the point image on the map.

The following image illustrates different markup drawings and their styles. You can edit the styles or add your own custom styles in the Module configuration.



Different types and styles of markup.

## Markup Measurement

Markup made by the [Measurement Module](#) can be edited and styled like any other markup. If edited, measurements are recalculated and re-drawn on the map showing the new measurements.



You can move, resize and rotate lines, polygons and text markup. Points can only be moved. Text can be edited.

If several drawings overlay each other, click the top-most drawing to place it in edit mode and to open the map tip. In the map tip, select the drawing you want to edit from the drop down list or use the next page or previous page arrows to go to the pages for the drawings.

## Export Drawings

The Export Drawings tool extracts the markup layer and creates an Export.zip file containing all of the drawings. This feature is enabled only after markup has been added to the map via the Viewer UI.

If the Edit Drawings tool is enabled, the Export Drawings tool will be disabled.

When the Export Drawings tool is selected, a confirmation dialog will appear. Once accepted, the Export.zip file is saved to the user's device.

### 15.45.0.1 Shapefiles

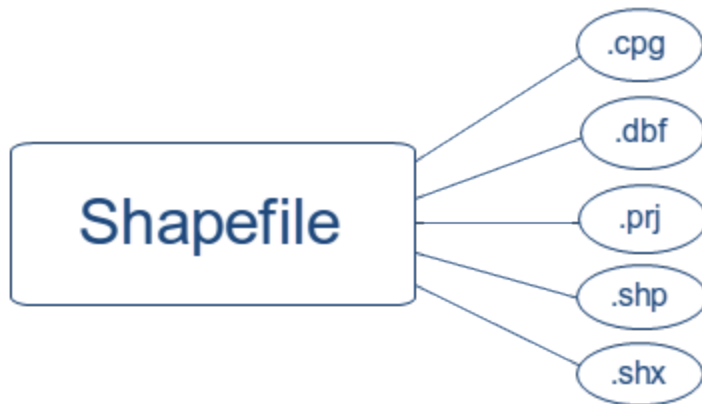
The shapefile format is a vector data format for GIS software developed by Esri. Shapefiles support point, line, and area features. A valid shapefile requires one .shp file as well as an .shx and .dbf file. For more information, see [Esri's technical specification for shapefiles](#).

### 15.45.0.2 Export.zip Contents

The Export.zip contains the user's drawings as shapefiles. Separate shapefiles are exported for each type of drawing. For example, if a map contained multiple point drawings and multiple polygon drawings, the export will contain a combined point.shp file and a combined polygon.shp file. Combined shapefiles can have the following names:


- point.shp
- polygon.shp
- polyline.shp
- multipoint.shp

In addition to a .shp file, the exported shapefiles consist of a .cpg, .dbf, .prj, and a .shx. All five of these files are required.



Exported shapefiles do not include measurement markup, although the geometry values and scale will be preserved. Similarly, exports do not include text or drawing styles from the markup layer.

## Copy to Drawing Layer

If a feature is selected and the panel displays its feature details, the user can use the Feature Actions Menu  to access the Copy to Drawing Layer action. This copies the geometry of the selected feature to the drawing layer. You can then interact with a markup version of the feature in the same way that you would interact with and edit other markup on the map.

## Commands

The Export Drawings feature can be configured to the I Want To menu with the `ExportMarkupLayer` command. For instructions, see [Configure the I Want To Menu on page 79](#). The `ExportMarkupLayer` command calls the `ExportGraphicsLayer` command, which exports any markup in the form of extents, multipoints, points, polylines, and polygons.

## Configuration Properties

### Module

None

### Views

The Markup Module does not have any views.

## View Models

- **MarkupViewModel:**
  - **markupLayerName:** The name of the markup layer as it appears in the list of map layers. The default name is **Drawings**.
  - **defaultPointMarkup:** The default style of points. You can change the style of points in the viewer using the **Styles** tool.
    - **pointStyle:** The default shape of points. Possible values include **Circle, Diamond, Square**.
    - **pointSize:** The default diameter of points in pixels.
    - **pointColor:** The default color of points in an ARGB hex string. All the colors in markup configuration use ARGB hex strings. For example, **#6600FF00** would be green with 40% opacity.
  - **defaultLineMarkup:** The default style of lines. You can change the style of lines in the viewer using the **Styles** tool.
    - **lineStyle:** The default style of lines. Possible values include: **Solid, Dot, Dash, Dash Dot, Dash Dot Dot**.
    - **lineWidth:** The default width of lines in pixels.
    - **lineColor:** The default color of lines in an ARGB hex string.
  - **defaultPolygonMarkup:** The default style of polygons. You can change the style of polygons in the viewer using the **Styles** tool.
    - **polygonBorderStyle:** The default style of the border of polygons. Possible values include **Solid, Dot, Dash, Dash Dot, Dash Dot Dot**.
    - **polygonFillStyle:** The default style of the fill of polygons. Possible values include **Solid, Cross, Diagonal Cross, Forward Diagonal, Backward Diagonal, Horizontal, Vertical**.
    - **polygonBorderWidth:** The default width of the border of polygons in pixels.
    - **polygonFillColor:** The default color of the fill of polygons in an ARGB hex string.
    - **polygonBorderColor:** The default color of the border of polygons in an ARGB hex string
  - **defaultTextMarkup:** The default style of text markup. You can change the style of text in the viewer using the **Styles** tool.
    - **textStyle:** The default style of text. Possible values include **Normal, Italic, Oblique**.
    - **textStyleWeight:** The default weight of text. Possible values include **Lighter, Normal, Bold, Bolder**.
    - **textSize:** The default size of text in points. For example, "14pt".



Text size is expressed as a string, not as a number.

- **textColor:** The default color of text in an ARGB hex string.
- **textFamily:** The default font family to use for text markup. You can use any HTML font-family style attribute. For example, you can use specific font family names such as **Arial** and **Segoe UI**, or use generic types such as **serif, sans-serif, cursive, fantasy, or monospace**. Font family is expressed as a string.
- **customMarkupTools:** If you create a custom tool that adds markup and you want the tool to work in the same way as the default tools, you must add the name of your custom tool to one of the four arrays in the **customMarkupTools** object.  
For example, if you want your custom tool to be able to change the style of the markup in the viewer, you



must add it to one of the arrays. If you add a custom tool to an array, for example a point array, and its style is changed in the viewer, it activates the point style picker so that you can change the style. The same is true for other styles:

- **point**: An array of strings containing the names of extra point markup tools.
  - **polyline**: An array of strings containing the names of extra polyline markup tools.
  - **polygon**: An array of strings containing the names of extra polygon markup tools.
  - **text**: An array of strings containing the names of extra text markup tools
- **StyleSelectorViewModel**: If you add a markup tool that is not part of the default tools, in order for the custom tool to work correctly, you must add the ID of the tool to one of the following arrays, depending on the type of markup your tool adds.
    - **customPointStyles**: An array of custom point tools.
    - **customLineStyle**: An array of custom line tools.
    - **customPolygonStyles**: An array of custom polygon tools.
    - **customTextStyles**: An array of custom text tools.
  - **TransientMarkupPaletteViewModel**: None

#### See Also...

[Configure the Toolbar on page 108](#)

[Measurement Module on page 269](#)

[Compact Toolbar Module on page 150](#)

[Tabbed Toolbar Module on page 363](#)

[Snapping Module on page 360](#)

## 15.46 Measurement Module

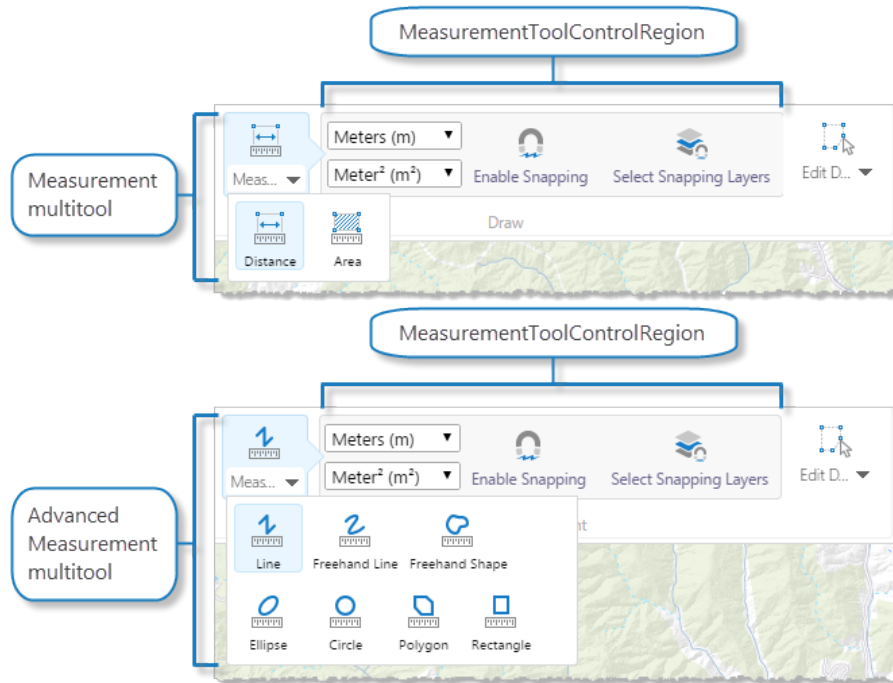


This module can be partially configured using Manager. For instructions, see [Configure Tool Behavior on page 121](#).

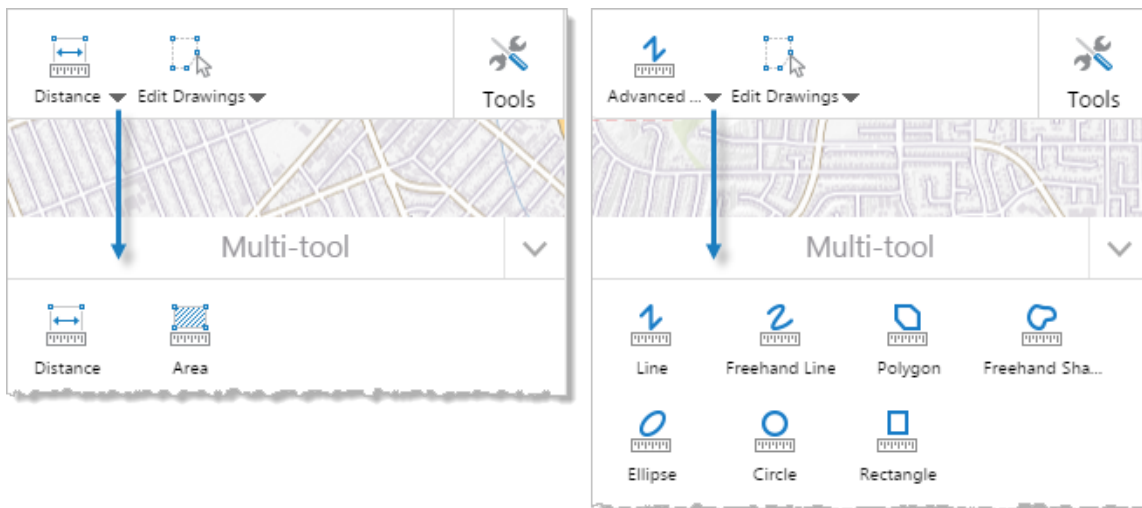
The Measurement Module implements the ability to measure distances and areas on the map.

For a user to be able to perform measurements, you must add one or more measurement tools to the toolbar.

Alternatively, you could add measurement tasks to the I Want To menu or create workflows that prompt the user to perform a measurement. The HTML5 Viewer provides a set of commonly-used measurement tools that you can add to the toolbar.



Measurement tools (top), and Advanced Measurement tools (bottom) in the Desktop and Tablet interfaces



Measurement tools (left), and Advanced Measurement tools (right) in the Handheld interface

## Measurement Tools

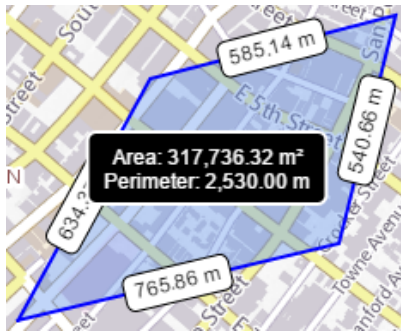
You can add measurement tools to a viewer's toolbar from the Toolbar section of the Management Pack. There are two measurement tool groups that you can activate:

- **Measurement Tool Group**
  - **Measure:** A multitool that adds measurement tools for distance and area.
  - **MeasurementToolControlRegion:** The context-sensitive toolbar for the Measure multitool. It adds a unit picker, snapping settings, and styling tools when the Measure tool is activated.
- **Advanced Measurement Tool Group**
  - **Measure:** A multitool that adds measurement tools for distance and area, including various shape tools.
  - **MeasurementToolControlRegion:** The context-sensitive toolbar for the Measure multitool. It adds a unit picker, snapping settings, and styling tools when the Measure tool is activated.

## Edit Measurement Drawings

The capability to include and provide editing of measurement drawings on a map tip is dependent on "enabled" set to true on the `GraphicsLayerIdentifyProvider` in the configuration of the Identify module. If "enabled" is set to false, the edit capability is not available.

When a user adds a measurement drawing to the map and then clicks the drawing, it is automatically placed in edit mode. The edit handles for the drawing are visible, and the map tip opens. The map tip includes the dimensions for the associated drawing.

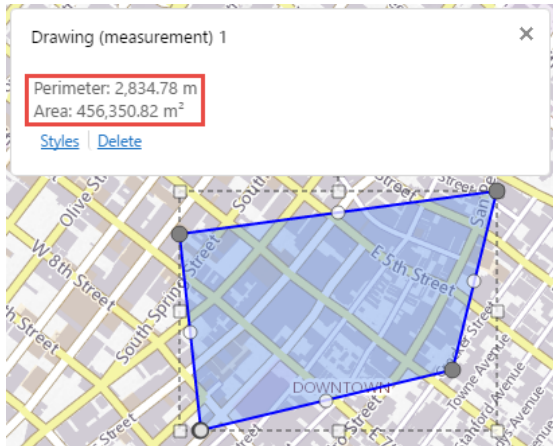


Area drawing on a map

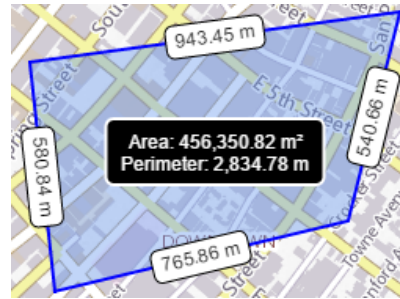


Area drawing in edit mode

When the user changes the length or shape of the drawing on the map, the dimensions change in the map tip.



Dimensions changed in the map tip



Dimensions shown on the drawing

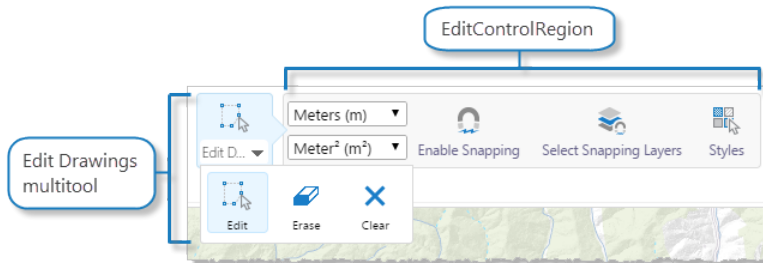
For information about editing the drawing on the map, changing its styles, and editing drawings that are overlaid, see [Edit Drawings on page 260](#) in the Markup Module.

Note that drawings and measurement drawings are included in the results of an identify operation.

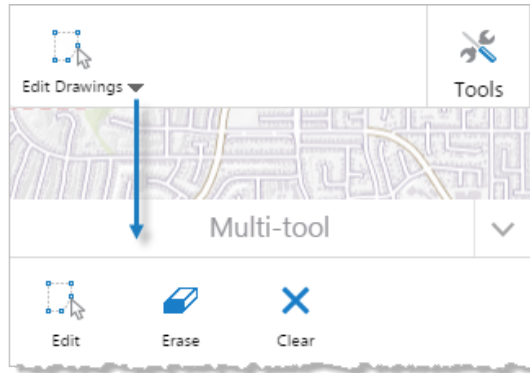
## Measurement Editing Tools

Clicking a drawing to access the editing function via the map tip is preferable to using the **Edit Drawings** multitool in the toolbar.

**Edit Drawings** allows the user to edit, erase, or clear drawings. Note that using **Erase** on drawings that overlay each other erases all of the drawings. To avoid this behavior, the user can click the graphic to open the map tip, select the drawing to be deleted from the drop down list, and click **Delete**.



Measurement editing tools in the Desktop and Tablet interfaces

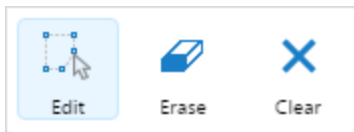


### Measurement editing tools in the Handheld interface

To add measurement editing tools, you can add the **Edit Drawings** tool group:

- **Edit Drawings Tool Group**
  - **Edit Drawings:** A multitool that allows you to edit existing map markup. It includes Edit, Erase, and Clear tools.
  - **EditControlRegion:** The context-sensitive toolbar for the Edit Drawings multitool. It adds a unit picker, snapping settings, and styling tools when the Edit Drawings tool is activated.
  - **Export Drawings:** A tool for exporting map markup. For more information, see [Export Drawings](#) and the [Markup Module](#).

If the Edit tool is selected, the Erase and Clear tools can be activated. These tools apply to both drawings and measurements.



With the Edit tool activated, the Erase, and Clear tools become available.

Note that using **Erase** on drawings that overlay each other results in erasing all of the drawings. To avoid this behavior, the user can click the graphic to open the map tip, select the drawing to be deleted from the drop down list, and click **Delete**.

## Measurement and Markup

When the user performs a measurement, the segment, perimeter, or area measurements display as markup on the map. The Measurement Module and the [Markup Module](#) are integrated so that measurement markup can be edited and styled like other markup.

When the user edits the length or area of measurement markup, the measurements are recalculated and adjusted on the map and in the map tip.



Measurements recalculated after the user changes the markup area

## Measurement Data and the Results Table

You can configure viewers to show measurement data in the Results Table as well as on the map. See [the setup procedure](#) for instructions. The Results Table provides an easy-to-read presentation of measurement data. In addition:

- The Results Table shows the angle and bearing, neither of which show on the map.
- The Results Table has an export function that enables the user to export the data to a file.

The Results Table opens when the user completes the first measurement. Each measurement appears on a new tab. If the measurement shape is made up of straight lines, the table contains one line for each segment of the measurement.

When the user selects a row in the table, the map pans to the shape that the row represents. It highlights the shape on the map and opens a map tip containing the data from that row.

Polygon 1	Line 1							
Segment	X	Y	Length	Units	Area	Area Units	Angle	Bearing
Total			67,131.20	m	219,973,900.46	m <sup>2</sup>		
1	-13,386,643.9837	4,029,065.4076	19,948.10	m			10.8099°	N 79.1901° E
2	-13,363,024.9420	4,033,575.1923	4,214.63	m			261.3844°	S 8.6156° W
3	-13,363,789.3123	4,028,530.3484	6,299.20	m			343.0245°	S 73.0245° E
4	-13,356,527.7946	4,026,313.6746	4,377.08	m			287.6501°	S 17.6501° E
5	-13,354,922.6170	4,021,268.8307	14,811.85	m			201.6191°	S 68.3809° W
6	-13,371,509.4521	4,014,695.2463	17,196.92	m			135.5968°	N 45.5968° W
7	-13,386,338.2356	4,029,218.2817	283.42	m			206.5651°	S 63.4349° W

**Measurement data in the Results Table**

Measurement map tips and layer map tips get the header and main content from different sources. Measurement map tips get their content from the measurements, not from the layer configuration. Both types of map tip use the same footer configuration. For more information, see [MapTips Module on page 253](#).

The user can export the measurement data to a file by selecting the Export Results option in the Panel Actions Menu, provided the Menu Module's ResultsTableActions has at least one export item configured. See [Menu Module on page 282](#) for more information.

**To show measurement results in the Results Table:**

1. In a text editor or XML editor, open one of the viewer configuration files—Desktop.json.js or Tablet.json.js.

By default, the configuration files are stored in the Sites folder:

```
Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default
```

2. Find the `Measurement` property in the `Results` Module's `resultMappings`.

```

...
{
  "moduleName": "Results",
  "moduleType":
"geocortex.essentialsHtmlViewer.mapping.infrastructure.results.ResultsModule",
  "libraryId": "Mapping.Infrastructure",
  "configuration": {
    "resultMappings": {
      ...
      "Measurement": [],
      ...
    }
  }
}

```

3. Set the `Measurement` property to `ShowResultsTable`.

```
"Measurement": ["ShowResultsTable"],
```

4. Save the file.
5. Repeat these steps for the other configuration file.

## Measurement Projection

You can configure measurement projection settings from the Tool Behavior section of the Management Pack or by setting the Measurement Module's configuration properties in the viewer configuration files. For more information about the Tool Behavior settings, see [Configure Tool Behavior on page 121](#).

**Measurement**

---

Projection WKID:

Default Length Unit:  ▼

Default Area Unit:  ▼

Calculation Type:  ▼

Add as Drawing:

Prediction Enabled:

### The measurement settings in the Tool Behavior section of the Management Pack

By default, projection is disabled. This setting only applies if the Calculation Type setting is set to Planar. If it is set to Geodesic or PreserveShape but `measurementProjectionWkid` is specified in the viewer configuration files, the viewer overrides the set calculation type and uses Planar.



If no measurement projection WKID is specified, the measurement calculation is done in the current spatial reference of the map. If no measurement projection WKID and no calculation type is specified, the viewer uses the geodesic calculation type, which does not require a WKID.

## Configuration Properties

### Module

- **tools**: An array specifying the measurement tools (if any) that need to be registered with the tool registry. Measurement tools have the following properties:

- **name**: The name that identifies this tool.



If your viewer is going to be available in more than one language, enter the text key that the tool name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **command**: The command to be invoked by this tool. The command is either `MeasureArea` or `MeasureDistance`.
- **drawMode**: The type of geometry that the user draws and the tool operates on. The possible modes include `POLYGON` for `MeasureArea` or `POLYLINE` for `MeasureDistance`.
- **isSticky**: When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.  
If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
- **iconUri**: The image to display for this tool.
- **statusText**: The status message to display when the tool is activated, often containing instructions for the user. You can use a text key or the literal text.



If the tools are being registered from the [TabbedToolbar Module](#), they do not need to be registered here.

- **measurementProjectionWkid**: Defines the WKID of the projection for the Measurement Module to use. If not defined, projection is disabled. The default setting is to leave the field undefined.
- **measurementLengthUnits**: Defines the default unit of measurement to use for length. Possible values include **feet**, **yard**, **meter**, **kilometer**, **mile**, and **nauticalMile**.
- **measurementAreaUnits**: Defines the default unit of measurement to use for area. Possible values include **sqFeet**, **sqYard**, **sqMeter**, **sqKilometer**, **sqMile**, **sqNauticalMile**, **acre**, and **hectare**.
- **coordinateFractionalDigits**: The number of digits to show after the decimal point for coordinates and angles. The default is 4.



This precision setting cannot exceed the maximum representable IEEE 754 floating point. This is specified in the IEEE 754 standard. See [double-precision floating-point format](#) and [significant figures](#) for more information.

- **measurementFractionalDigits**: The number of digits to process after the decimal point for measurements. The default is 2.



This precision setting cannot exceed the maximum representable IEEE 754 floating point. This is specified in the IEEE 754 standard. See [double-precision floating-point format](#) and [significant figures](#) for more information.

- **degreeFormat**: The format for displaying angles and bearing. Possible values are **dd** (decimal degrees), **ddm** (degrees/decimal minutes), or **dms** (degrees/minutes/seconds).
- **angleDirectionSystem**: The direction system for angles. Possible values are **polar**, **north\_azimuth**, or **south\_azimuth**. For information about direction systems, refer to [About direction measuring systems and units](#) in the ArcGIS documentation.
- **measurementResultTypes**: A list of measurement shapes that generate measurement results. The HTML5 Viewer supports the shapes listed [here](#).
- **enablePrediction**: Defines whether or not measurement units are predicted as you begin typing a unit. When set to `false`, it disables predictive measurements. The default value is `true`.
- **calculationType**: Defines the calculation used by the Measurement Module. Only one of three values apply—`preserveShape`, `geodesic` or `planar`.

## Views

- **MeasurementView**: (Desktop and Tablet interfaces only) None
- **MeasurementUnitSwitcherView**: (Handheld interface only) None

## View Models

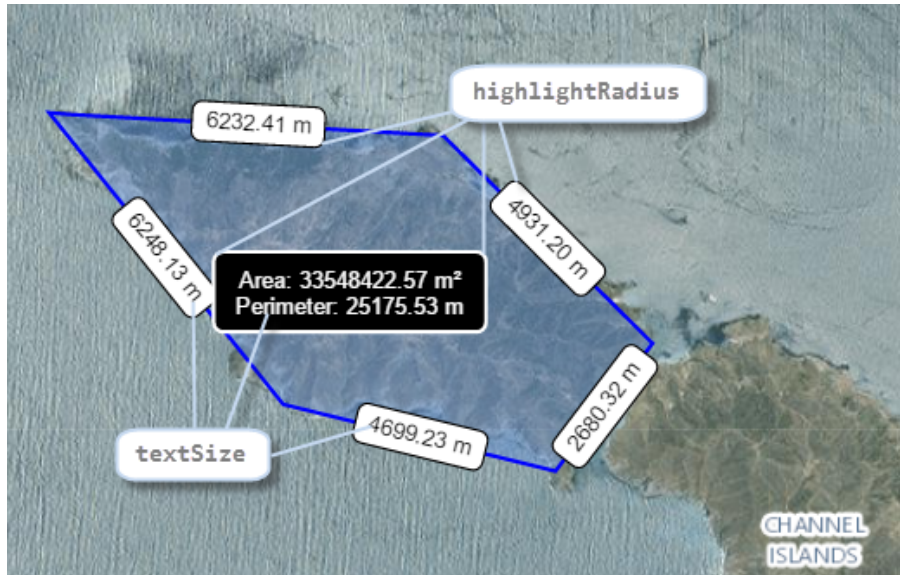
- **MeasurementViewModel**:
  - **markupLayerName**: The name of the layer to add the markup to. The name is used in the viewer's list of map layers. The default name is **Drawings**.



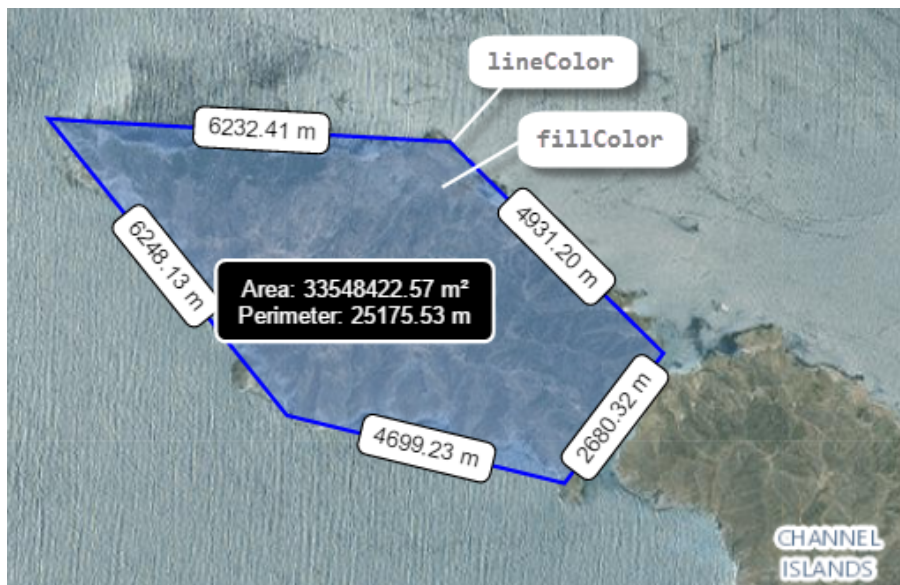
In order for the markup drawn by the [Measurement Module](#) to be editable by the Markup Module, the `markupLayerName` must be the same for both modules.

- **addMarkupToMapByDefault**: Defines whether markup remains on the map until the user's session ends. The default value is `true`.  
When this property is set to `false` the markup is removed after the user performs another action.

- **Text Size, Corner Rounding:** The following properties control text size and corner rounding:

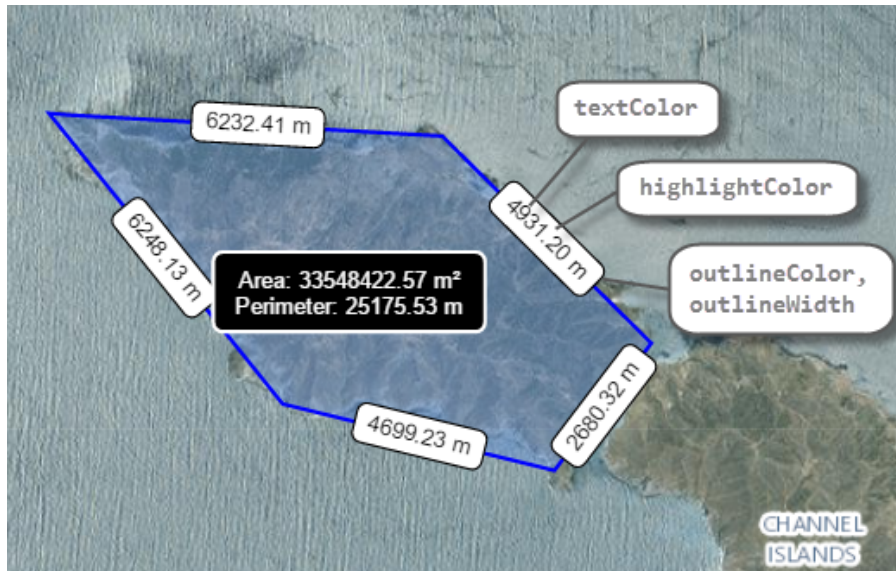


- **textSize:** The size of the text markup for line segments and measurement totals, in pixels. You must specify the units. The default is **12px**.
- **highlightRadius:** The radius to round the corners of the boxes that enclose line segment measurements and measurement totals, in pixels. When you specify the radius, omit the units. The default radius is **5**.
- **Measurement Colors:** The following properties control the colors of measurement lines and areas:



- **lineColor:** The color of measurement lines on the map, including the lines for distance measurements and the outlines of area measurements. The color can be an HTML color name or a hexadecimal value, for example, **red** or **#FF0000**. The default color is **#0000FF** (blue).

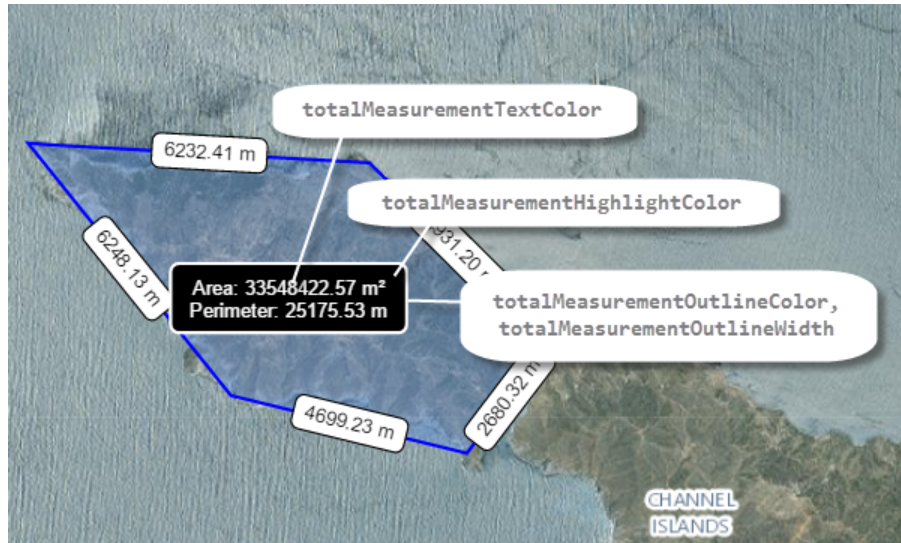
- **fillColor:** The fill color inside area measurements. The color can be an HTML color name or a hexadecimal value, for example, **green** or **#00FF00**. The default color is **#6495ED** (cornflower blue).
- **Line Segment Measurements:** The following properties control the appearance of the markup for the individual line segments in distance and area measurements:



- **textColor:** The color of the markup text for the individual line segments in a measurement. The color can be an HTML color name or a hexadecimal value, for example, **blue** or **#0000FF**. The default color is **#000000** (black).
- **highlightColor:** The fill color for the boxes that enclose line segment measurements. The color can be an HTML color name or a hexadecimal value, for example, **red** or **#FF0000**. The default color is **#FFFFFF** (white).
- **outlineColor:** The border color for the boxes that enclose line segment measurements. The color can be an HTML color name or a hexadecimal value, for example, **red** or **#FF0000**. The default color is **#000000** (black).
- **outlineWidth:** The border width for the boxes that enclose line segment measurements, in pixels. When you specify the width, omit the units. The default width is **1**.
- **lineDescriptionFormat:** The text content for each line segment. You can include up to three different tokens, each representing different aspects of the line segment:
  - **{0}**: The distance of the line. For example, a line that is 100 meters in length displays as **100.00 m**.
  - **{1}**: The polar angle of the line. For example, a line heading directly east displays as **0.0000°**. A line heading directly north displays as **90.0000°**.
  - **{2}**: The quadrant bearing of the line. For example, a line heading 20 degrees east of north displays as **N 20.0000° E**.

By default, this property is omitted, which is identical to specifying a value of **{0}**.

- **Total Measurement:** The following properties control the appearance of the markup for measurement totals:



- **totalMeasurementTextColor:** The color of the markup text for measurement totals. The color can be an HTML color name or a hexadecimal value, for example, **red** or **#FF0000**. The default color is **#FFFFFF** (white).
- **totalMeasurementHighlightColor:** The fill color for the box that encloses a measurement total. The color can be an HTML color name or a hexadecimal value, for example, **red** or **#FF0000**. The default color is **#000000** (black).
- **totalMeasurementOutlineColor:** The border color for the box that encloses a measurement total. The color can be an HTML color name or a hexadecimal value, for example, **red** or **#FF0000**. The default color is **#FFFFFF** (white).
- **totalMeasurementOutlineWidth:** The border width for the box that encloses a measurement total, in pixels. When you specify the width, omit the units. The default width is **2**.

#### See Also...

[Configure Tool Behavior](#) on page 121

[Configure the I Want To Menu](#) on page 79

[Snapping Module](#) on page 360

[MapTips Module](#) on page 253

[Markup Module](#) on page 258

[Results Module](#) on page 328

[TabbedToolbar Module](#) on page 363

[CompactToolbar Module](#) on page 150

[About User Interface Text](#) on page 64

## 15.47 Menu Module

The Menu Module implements menus and menu-like components that are used by other modules. For example, a `FeatureActions` menu is used by the [FeatureDetails Module](#). The menu and its items are configured in the Menu Module. A menu item runs a configured `command` with a configured `commandParameter` as its argument.



You can configure menus from the Management Pack. See [Configure the Context Menus on page 81](#) for more information.

### Configuration Properties

#### Module

- **menus:** An array of menus that belong to one or more other modules:
  - **id:** The menu's ID.
  - **description:** A short description of the menu.



If your viewer is going to be available in more than one language, enter the text key that the description is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **moduleId:** The ID of the module that the menu belongs to.
- **defaultIconUri:** The URI of the default icon for menu items.
- **items:** An array of menu items. Menu items have the following properties:
  - **iconUri:** The image to display beside the menu item. The recommended image size is 24px by 24px.
  - **text:** The text to appear on the menu item. You can use a text key or the literal text.
  - **description:** A brief description of the menu item to appear below the `text`. You can use a text key or the literal text.
  - **command:** The command to execute when the menu item is selected.



If you set the `command` property, do not set the `batch` property.

- **commandParameter:** The parameter value to pass to the command when it runs, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.





If you set the `commandParameter` property, do not set the `batch` property.

In the following example from the **FeatureActions** menu, a **commandParameter** is passed into the **ShowFeatureDetailsCompact** command to display feature details in the sidebar:

```
"command": "ShowFeatureDetailsCompact",
"commandParameter": "{{context}}"
```

The **{{context}}** token is a special token that retrieves the default menu context which a menu may have. For example, the default context of the **FeatureActions** menu is the current *feature*. Therefore, the command displays feature details about the current *feature*.

- **hideOnDisable**: If this property is set to `true` and the menu item's command cannot run, the menu item does not show in the menu. If `hideOnDisable` is `false` and the menu item's command cannot run, the menu item shows in the menu, but it is grayed out.
- **batch**: An array of objects, each with three properties: an executable command, an optional command parameter, and an optional Boolean to abort the execution of further commands if a command fails to execute. This allows multiple commands to be executed in a specified order, although some commands may trigger asynchronous actions.

The following example attempts to activate the Tabbed Toolbar and, if successful, proceeds to greet the user with an alert:

```
"batch": [
  {
    "command": "ActivateView",
    "commandParameter": "TabbedToolbarView",
    "abortBatchOnFailure": true
  },
  {
    "command": "Alert",
    "commandParameter": "Hello!"
  }
]
```

If omitted, `abortBatchOnFailure` is `false` by default.



If you set the `batch` property, do not set the `command` or `commandParameter` properties.

## Views

The Menu Module does not have any views.

## View Models

The Menu Module does not have any views models.

### See Also...

[Editing Module on page 160](#)

[FeatureDetails Module on page 168](#)

[SimpleLayerList Module on page 358](#)

[SkipLinks Module on page 359](#)

## 15.48 NativeIntegration Module



Prior to HTML5 Viewer 2.3, the NativeIntegration Module was called the Native Module.

The NativeIntegration Module implements platform-specific functionality like the File Attachments feature. The File Attachments feature uses the HTML5 File Reader to attach a file or photograph to a feature in a feature layer. The File Attachments feature then automatically appears whenever the browser supports this functionality. No additional configuration is required.



In order to attach files to a feature, the feature layer must be editable and it must support attachments.

The following browsers support the File Attachment feature:

- Internet Explorer 10 and later
- Firefox 3.6+
- Chrome 6+

The following browsers support File Attachments on Tablets:

- Safari iOS 9.0+
- Android browser 4.0+

The following browsers support File Attachments on Mobile Phones:

- Android 4.0+
- Safari iOS 9.0+
- Chrome for Android (Android 4)



To see which other browsers support the HTML5 File Reader, see [caniuse.com/#feat=filereader](http://caniuse.com/#feat=filereader).





If the browser on a mobile device does not support HTML5 FileReader, then the File Attachment feature attempts to use the older Cordova environment that natively supports it. The Cordova environment is not available for desktop machines.

## Module

None

## Views

- **AttachFileView:** None

## View Models

- **AttachFileViewModel:**
  - **AttachFileViewId:** The ID of the view in the NativeIntegration Module that displays file attachments.

## 15.49 Navigation Module

The Navigation Module determines the order in which navigation controls appear, including the geolocation, zoom in, zoom out, and bookmarks controls.



### Navigation controls on the map in the Desktop interface (left), and Handheld interface

- **Geolocate:** Opens the Geolocation Options menu, which offers to either find, track or follow the user's location. For more information, see the [Geolocate Module](#). By default, this button appears at the top left of the screen in the Desktop and Tablet interfaces, and the bottom left of the screen in the Handheld interface.



If your map is not in Web Mercator or WGS 84, you must configure an ArcGIS geometry service so the geolocation widget can project from latitude/longitude to the coordinates of your map. For instructions on configuring a geometry service, see [Configure Application-Wide Settings on page 70](#).

- **Zoom in:** Zooms in the map. This button is hidden by default in the Handheld interface.
- **Zoom out:** Zooms out the map. This button is hidden by default in the Handheld interface.
- **Bookmarks Button:** Opens the Bookmarked Locations menu, which allows you to create and remove bookmarks, and jump to a bookmarked location.



To display Bookmarks button, configure the [Bookmarks Module](#).

## Configuration Properties

### Module

None

### Views

- **DataFrameButtonView:** None
- **GeolocateButtonView:** None



GeolocateView itself has moved to the [Geolocate Module](#).

- **ZoomInView:** None



In the Handheld interface, this view is hidden by default. To display this view, set its `visibility` property to `true`.

- **ZoomOutView:** None



In the Handheld interface, this view is hidden by default. To display this view, set its `visibility` property to `true`.

- **BookmarksButtonView:** None



To display Bookmarks button, configure the [Bookmarks Module](#).

View Models

None

**See Also...**

[Bookmarks Module on page 138](#)

[Geolocate Module on page 184](#)

[ZoomControl Module on page 395](#)

## 15.50 Offline Module



This module must be configured using Manager—you cannot configure the Offline feature by editing the configuration files. See [Configure a Viewer for Offline Use on page 94](#) for instructions.



From version 2.0 of the Geocortex Viewer for HTML5, offline support requires the Geocortex Mobile App Framework. To download and install the Geocortex Mobile App Framework, visit the [Geocortex Support Center](#).

The Offline Module supports [offline](#) functionality, which allows the end user to run the viewer and interact with the map, including editing features, without being connected to a network.

Offline editing depends on the [FeatureLayer Module](#), which implements support for feature layers in the HTML5 Viewer, and the [Editing Module](#), which implements editing of features. In order for the user to edit features when the viewer is offline, the features must belong to a feature layer that has editing enabled.

### Configuration Properties

Module

None

Views

None

View Models

None

**See Also...**

[Offline on page 401](#)

[Editing Module on page 160](#)

[FeatureLayer Module on page 176](#)

[Offline Maps Module on page 288](#)

## 15.51 Offline Maps Module



From version 2.0 of the Geocortex Viewer for HTML5, offline support requires the Geocortex Mobile App Framework. To download and install the Geocortex Mobile App Framework, visit the [Geocortex Support Center](#).


The Offline Maps Module implements the user interface for using and managing Offline Maps. For more information, see [Offline on page 401](#).


### Configuration Properties

#### Module

- **aoiMask**: An object representing the area outside the offline map area of interest, with the following properties:
  - **enabled**: If you want a visual indicator for the area beyond the offline map area, set to `true`; otherwise, set to `false`. The default is `true`.
  - **fillColor**: The color of the area beyond the offline map area, specified as an HTML color. The default is `null`, that is, no fill color.
  - **boundaryColor**: The color of the boundary outline between the offline map area and beyond it. The default is `black`.
  - **boundaryWidth**: The width of the boundary outline between the offline map area and beyond it. The default is `2`.
  - **boundaryStyle**: The style of the boundary outline between the offline map area and beyond it. Possible values include: `solid`, `dash`, `dot` and `null`. The default is `dash`.
  - **layerId**: The ID of the layer to use for the area outside the offline map area of interest. The default is `OfflineMapsAOIMask`.
  - **matchMapBackground**: If you want the map background color to match the fill color of the area beyond the offline map area, set to `true`; otherwise, set to `false`. This setting applies to all areas lacking map data. This setting only applies if the `fillColor` is not `null`. The default is `false`.

#### Views

- **ListOfflineMapView**:
  - **sync**: Determines the command to execute when a user taps the **Sync** button 
    - **command**: The command to execute. The default is `StartOfflineSyncAndShowProgress`.
    - **commandParameter**: A command parameter object with a single property:
      - **offlineMap**: The Offline Map to sync. By default, the `{{offlineMap}}` token is used. This special token represents the Offline Map selected.

- **download:** Determines the command to execute when a user first taps an Offline Map  to download it:
  - **command:** The command to execute. The default is `StartOfflineDownloadAndShowProgress`.
  - **commandParameter:** A command parameter object with a single property:
    - **offlineMap:** The Offline Map to sync. By default, the `{{offlineMap}}` token is used. This special token represents the Offline Map selected.
- **ManageOfflineMapView:**
  - **newOfflineMap:** Determines the command to execute when a user begins to create a new Offline Map:
    - **command:** The command to execute. The default is `ShowOfflineMapEditor`.
    - **commandParameter:** A command parameter object. By default, this object is empty.
  - **editOfflineMap:** Determines the command to execute when a user begins to edit an Offline Map:
    - **command:** The command to execute. The default is `ShowOfflineMapEditor`.
    - **commandParameter:** A command parameter object with a single property:
      - **existingOfflineMap:** The Offline Map to edit. By default, the `{{offlineMap}}` token is used. This special token represents the Offline Map selected.
- **OfflineMapEditorView:**
  - **title:** Determines the panel title:
    - **forNew:** The panel title to use when creating a new Offline Map. The default is `@language-offlinemapeditor-new-title`.
    - **forExisting:** The panel title to use when editing an existing Offline Map. The default is `@language-offlinemapeditor-title`.
- **OfflineMapEditorNameView:** None.
- **OfflineMapEditorLayersView:** None.
- **OfflineMapEditorGeometryView:**
  - **geometrylayerId:** The ID of the geometry layer to use for selecting an offline map area. The default is `OfflineMapEditorGeometry`.
  - **fillColor:** The fill color of the selected offline map area in RGBA notation. The default is `rgba(200, 0, 0, 0.3)`.
  - **outlineColor:** The outline color of the selected offline map area in RGBA notation. The default is `rgba(200, 0, 0, 0.7)`.
  - **outlineWidth:** The outline width of the selected offline map area in pixels. The default is 1.

- **OfflineMapEditorBasemapsView:**
  - **showBasemapsLevelEditorView:** Determines the command to execute when a user begins to select the maximum level of detail for the Offline Basemap:
    - **command:** The command to execute. The default is `ShowOfflineMapEditorBasemapLevels`.
    - **commandParameter:** A command parameter object with a single property:
      - **basemap:** The Offline Basemap for which to select the maximum level of detail. By default, the `{{basemap}}` token is used. This special token represents to the Offline Basemap selected.
- **OfflineMapEditorBasemapLevelsView:** None.
- **OfflineMapEditorSharingView:** None.
- **SyncStatusView:**
  - **showXButton:** To display a button to close the Sync dialog window, set to `true`; otherwise set to `false`. The default is `false`.

## View Models

- **OfflineMapsViewModel:**
  - **saveOfflineMaps:** To allow users to save Offline Maps, set to `true`; otherwise set to `false`. The default is `true`.
  - **defaultThumbnail:** The path to the default Offline Map preview thumbnail.
  - **menus:** An array of menus. By default, these include the `ListOfflineMapsActions` and `ManageOfflineMapsActions` menus. Each menu has the following properties:
    - **id:** The menu's ID.
    - **description:** A short description of the menu.



If your viewer is going to be available in more than one language, enter the text key that the description is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **moduleId:** The ID of the module that the menu belongs to.
- **defaultIconUri:** The URI of the default icon for menu items.
- **items:** An array of menu items. Menu items have the following properties:
  - **iconUri:** The image to display beside the menu item. The recommended image size is 24px by 24px.
  - **text:** The text to appear on the menu item. You can use a text key or the literal text.
  - **description:** A brief description of the menu item to appear below the `text`. You can use a text key or the literal text.

- **command:** The command to execute when the menu item is selected.



If you set the **command** property, do not set the **batch** property.

- **commandParameter:** The parameter value to pass to the command when it runs, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.



If you set the **commandParameter** property, do not set the **batch** property.

In the following example from the **FeatureActions** menu, a **commandParameter** is passed into the **ShowFeatureDetailsCompact** command to display feature details in the sidebar:

```
"command": "ShowFeatureDetailsCompact",
"commandParameter": "{{context}}"
```

The **{{context}}** token is a special token that retrieves the default menu context which a menu may have. For example, the default context of the **FeatureActions** menu is the current **feature**. Therefore, the command displays feature details about the current **feature**.

- **hideOnDisable:** If this property is set to **true** and the menu item's command cannot run, the menu item does not show in the menu.  
If **hideOnDisable** is **false** and the menu item's command cannot run, the menu item shows in the menu, but it is grayed out.
- **batch:** An array of objects, each with three properties: an executable command, an optional command parameter, and an optional Boolean to abort the execution of further commands if a command fails to execute. This allows multiple commands to be executed in a specified order, although some commands may trigger asynchronous actions.

The following example attempts to activate the Tabbed Toolbar and, if successful, proceeds to greet the user with an alert:

```
"batch": [  
  {  
    "command": "ActivateView",  
    "commandParameter": "TabbedToolbarView",  
    "abortBatchOnFailure": true  
  },  
  {  
    "command": "Alert",  
    "commandParameter": "Hello!"  
  }  
]
```

If omitted, `abortBatchOnFailure` is `false` by default.



If you set the `batch` property, do not set the `command` or `commandParameter` properties.

- **OfflineMapEditorViewModel:**

- `thumbnail:`
  - `width:` The width of the Offline Map preview thumbnail in pixels. The default is 70.
  - `height:` The width of the Offline Map preview thumbnail in pixels. The default is 50.
  - `dpi:` The thumbnail resolution in Dots Per Inch. The default is 20.
- `steps:` An array of views that represents the steps for creating and editing Offline Maps. By default, they include: `OfflineMapEditorNameView`, `OfflineMapEditorGeometryView`, `OfflineMapEditorLayersView`, `OfflineMapEditorBasemapsView` and `OfflineMapEditorSharingView`.
- `onEditingStarted:` An array of commands with parameters to execute when a user begins editing an Offline Map. By default, a single command is included:
  - `command:` The command to execute. The default is `DeactivateView`.
  - `commandParameter:` The parameter to pass to the command. The default is `LayerDataContainerView`.



- **onEditingFinished**: An array of commands with parameters to execute when a user finishes editing an Offline Map. By default, a two commands are included:
  - **command**: The command to execute. Both default commands are `ActivateView`.
  - **commandParameter**: The parameter to pass to the command. The first default command parameter is `LayerDataContainerView`; the second default command parameter is `ManageOfflineMapView`.
- **tools**: An array of tools with which the user can select the map area to use offline. By default, the tools included are: Current Extent, Polygon, Freehand Polygon, Circle, Ellipse and Rectangle. Each tool can have the following properties:
  - **id**: A unique ID for this `tool`.
  - **type**: The type is `tool`.
  - **iconUri**: The URI for the icon that you want to appear on the tool. The image must be an appropriate size to fit on the tool. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **command**: The command that the tool runs after the user has drawn the geometry for the command to operate on.  
For information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
  - **drawMode**: The type of geometry the user draws, upon which the tool operates.
  - **name**: The name that you want to appear on the tool. You can use a text key or the literal text. For example, `@language-toolbar-tasks-identify` or **Identify**.
  - **tooltip**: The text for the tool tip that opens when the user positions the pointer over the tool. You can use a text key or the literal text. For example, `@language-toolbar-identify-point-tooltip` or **Find out about a location on the map**.
  - **hideOnDisable**: If this property is set to `true` and the tool's command cannot run, the tool does not show in the toolbar. If `hideOnDisable` is `false` and the tool's command cannot run, the tool shows in the toolbar, but it is grayed out.
  - **isSticky**: When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time. If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
  - **statusText**: The status message to display when the tool is activated, often containing instructions for the user. You can use a text key or the literal text. For example, `@language-toolbar-identify-point-desc` or **Click or tap a location on the map to learn what's there**.

- **menus**: An array of menus. By default, this includes a single menu: `OfflineMapEditorGeometryActions`. Each menu has the following properties:
  - **id**: The ID of the menu. The default is `OfflineMapEditorGeometryActions`.
  - **description**: A short description of the menu.



If your viewer is going to be available in more than one language, enter the text key that the description is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **items**: An array of menu items to select the map area to use offline. By default, the menu items included are: Clear, Current Extent, Polygon, Freehand Polygon, Circle, Ellipse and Rectangle. Menu items have the following properties:
  - **text**: The text to appear on the menu item. You can use a text key or the literal text.
  - **description**: A brief description of the menu item to appear below the `text`. You can use a text key or the literal text.
  - **iconUri**: The image to display beside the menu item. The recommended image size is 24px by 24px.
  - **command**: The command to execute when the menu item is selected.
  - **commandParameter**: The parameter value to pass to the command when it runs, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.

- **OfflineMapEditorBasemapLevelsViewModel**: None.
- **SyncStatusViewModel**: None.

#### See Also...

[Offline on page 401](#)

[Offline Module on page 287](#)

## 15.52 OptimizerIntegration Module

Geocortex Optimizer is a product that collects and reports on a GIS infrastructure. The Geocortex Viewer for HTML5 allows administrators to collect information about users. The OptimizerIntegration Module then collects data about:

- Viewer users
- Map areas viewed
- Tools used

This data is collected and can be used to create informative reports within Geocortex Optimizer.

## Configuration Properties

### Module

- **enabled:** If `true`, then the Viewer sends data to Optimizer. The default value is `false`.
- **userName:** The name of the user.
- **dataRelayUri:** The URI to the Optimizer database REST endpoint. If the endpoint is not specified, the OptimizerIntegration Module attempts to log back to the same host that the Viewer application is launched from.

### See Also...

[InsightIntegration Module on page 198](#)

## 15.53 OverviewMap Module

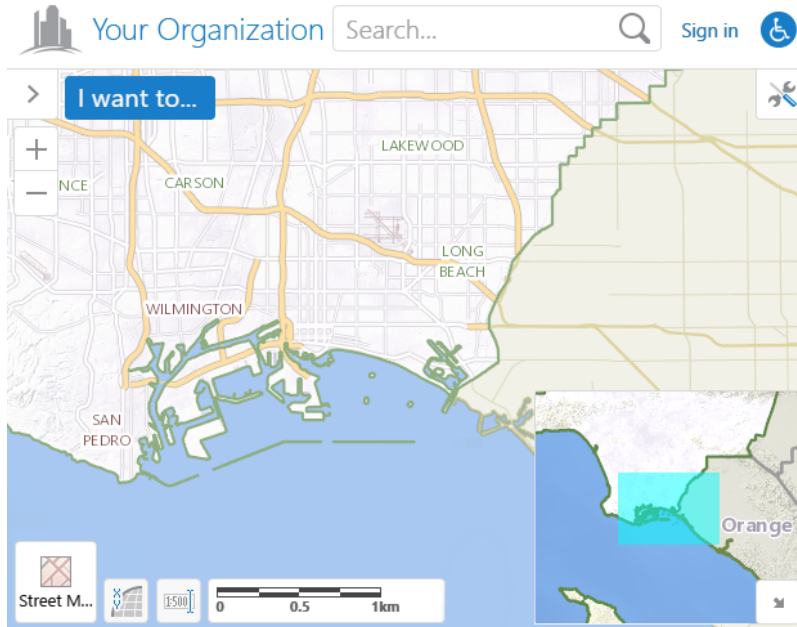


An Overview Map is not recommended for most modern applications, as it is easy to zoom out for quick context. One exception is a map that opens to a specific feature of interest—for example, a property parcel—from a third-party system, where the user lacks any idea of where the feature is located and has no other reason to zoom out.



This module can be configured using Manager. See [Configure Map Widgets on page 90](#) for instructions.

The OverviewMap Module displays the overview map that provides geographical context of the current map extent. The user can navigate the map by dragging around the rectangle in the overview map that represents the current map extent. The user can also open or close the overview map at will.



Overview map example

## Configuration Properties

### Module

- None

### Views

- **OverviewMapView:** None

### View Models

- **OverviewMapViewModel:**
  - **isEnabled:** To enable the overview map feature, set to `true`; otherwise, set to `false`. The default is `true`.



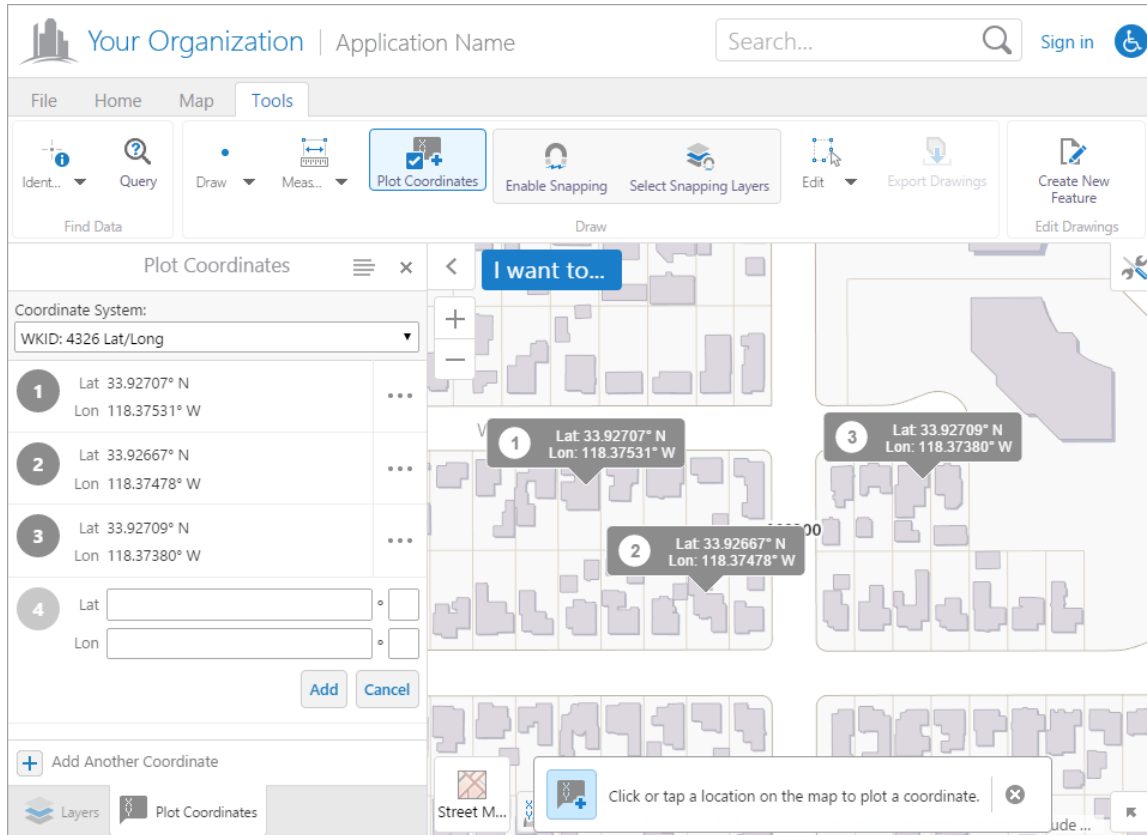
An overview map will only appear if one is configured in your site. For more information, see "The Overview Map" in the *Geocortex Essentials Administrator Guide*.

- **openByDefault:** To open the overview map when the viewer starts, set to `true`; otherwise, set to `false`. The default is `false`.
- **extentScaleFactor:** The scale factor to use for the overview map in relation to the current map extent. The default is 2.
- **visibleExtentColour:** A valid HTML color to use for the rectangle that represents the current map extent. The default is `#00FFFF`.

## 15.54 PlotCoordinates Module

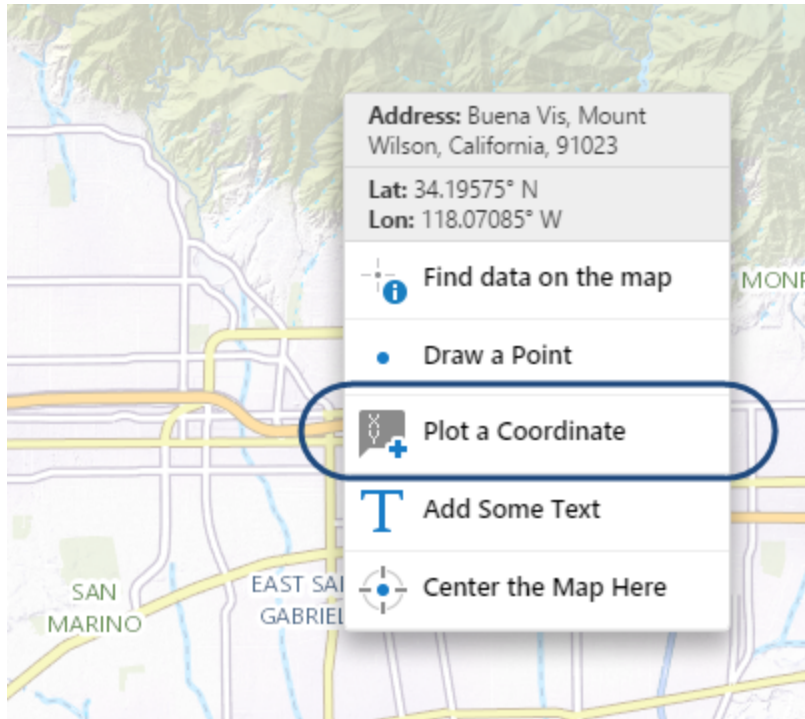
The PlotCoordinates Module implements the ability for end users to show the coordinates for specific locations on the map. To show the coordinates for a particular location, the user can either select the location on the map or input coordinates using text boxes in the Plot Coordinates panel.

Plotted coordinates can be printed, saved as part of a project, or saved to an ArcGIS web map. For more information, see [Printing Module](#), [Project Module](#), and [ExportWebMap Module](#). Plotted coordinates are removed from the map when the user ends the session.



Coordinates plotted on the map, shown in the Desktop interface

Alternatively, the user can right-click the map and select the **Plot a Coordinate** option.

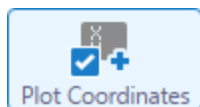


The right-click context menu with the Plot a Coordinate option.



On a handheld device, the user must open the Plot Coordinates panel to input coordinates via the input text boxes.

In order for users to plot coordinates on the map, you must provide a way for them to open the Plot Coordinates panel. The HTML5 Viewer provides a **Plot Coordinates** tool that you can add to the toolbar. The toolbar can be configured using Manager. See [Configure the Toolbar on page 108](#) for instructions.



### Plot Coordinates tool


Alternatively, you can add an item to the I Want To menu or create a workflow that invokes the `ActivatePlotCoordinatesView` command to open the panel. This module can be configured using Manager. For instructions, see [Configure the I Want To Menu on page 79](#).

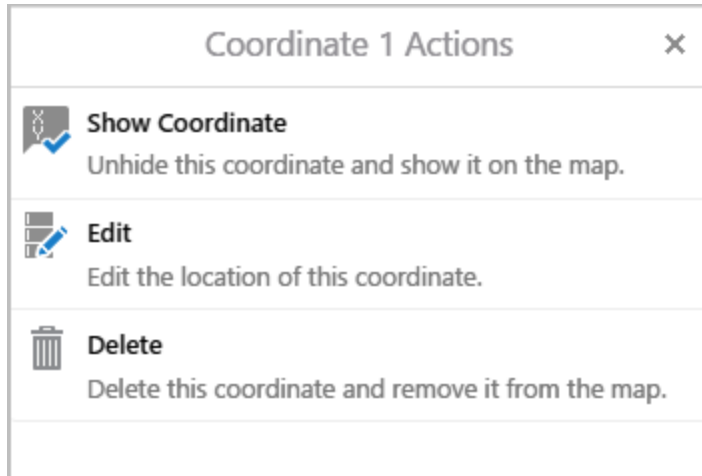
Plotted coordinates exist only for the duration of the session—when the user closes the viewer, the coordinates disappear. To save plotted coordinates across sessions and share it with other users, a signed-in user can save a project, provided the viewer supports projects. For information about projects, see [Project Module on page 304](#).

## Change the Coordinate System

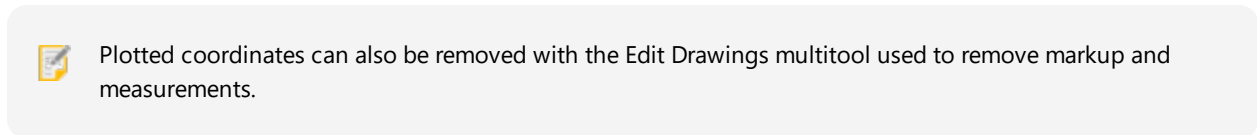
While the Plot Coordinates panel is open, the active coordinate system can be changed with the **Coordinate System** drop-down menu.

## Show, Hide, Edit, or Remove Coordinates

The user can choose to show, hide, edit, or remove any saved plotted coordinate from a list of available actions by selecting the  icon next to a coordinate in the Plot Coordinates panel.



The Plot Coordinates actions for a hidden coordinate. If the coordinate were shown, an action to Hide Coordinate would be included in this menu in the place of Show Coordinate.




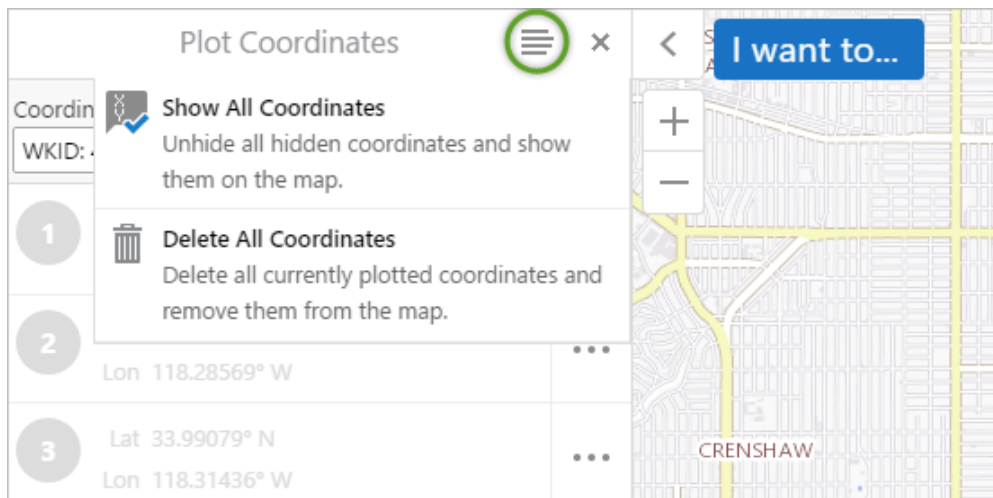
The icon and the callout map tips on the map indicate that they are being edited by turning blue. The user can select a new location on the map or edit the coordinates using the text boxes and selecting the **Update** button.



A Plot Coordinates callout map tip being edited (left) and a Plot Coordinates callout map tip (right)

### Show, Hide or Delete All Coordinates

The user can show or hide all coordinates with the Show All Coordinates, Hide All Coordinates, and Delete All Coordinates actions available from the Plot Coordinates panel's Action Menu. This menu is accessed from the  icon at the top of the panel.



The Show All Coordinates and Delete All Coordinates actions, displayed when the Coordinates Panel Action Menu button is toggled.

## Display Plotted Coordinates in Results

The plotted coordinates can be configured to display in either the Results Table or the Results List by adding the command `ShowResultsTable` or `ShowResultsList` to the `coordinates` array in the [Results Module](#) configuration. This allows users to export coordinates.

## Configuration Properties

### Module

- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **CoordinateListItemClickedBehavior:** A behavior that runs an array of commands when a coordinate in the coordinates list is clicked. By default, this includes three commands: `ClearDefaultHighlights`, `PanToFeature`, and `HighlightFeatureDefault`.
  - **CoordinateEditingStartedBehavior:** A behavior that runs an array of commands when a coordinate starts to be edited. By default, this includes two commands: `ActivatePlottedCoordinatesView` and `OpenDataFrame`.
  - **CancelEdtCoordinateBehavior:** A behavior that runs an array of commands when a user discards an edit to a coordinate. By default, this includes one command: `ClearActiveTool`.
  - **CoordinateEditedBehavior:** A behavior that runs an array of commands while a coordinate is being edited. By default, this includes three commands: `ClearDefaultHighlights`, `ActivatePlotCoordinatesView`, and `OpenDataFrame`.
  - **CoordinateDeletedBehavior:** A behavior that runs an array of commands when a coordinate is deleted. By default, it includes one command: `ClearDefaultHighlights`.



- **CoordinateAddedBehavior:** A behavior that runs an array of commands when a coordinate is added. By default, it includes three commands: `ClearDefaultHighlights`, `ActivatePlotCoordinatesView`, and `OpenDataFrame`.
- **tools:**
  - **name:** The name of the tool.



If your viewer is going to be available in more than one language, enter the text key that the tool name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **command:** The command that the tool runs.  
For a list of commands, see the [Geocortex SDK for HTML5 API Reference](#).
- **drawMode:** The type of geometry the user draws, upon which the tool operates.
- **hideOnDisable:** If this property is set to `true` and the button's command cannot run under the current configuration or run-time conditions, the button does not show in the toolbar.
- **isSticky:** When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.  
If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
- **iconUri:** The image that displays beside the tool.
- **statusText:** The status message to display when the tool's input method is via mouse, often containing instructions for the user. You can use a text key or the literal text.
- **keyboardStatusText:** The status message to display when the tool's input method is via keyboard, often containing keyboard shortcut hints for the user. You can use a text key or the literal text.

## Views

- **PlotCoordinatesView:** None
- **CoordinateActionsView:**
  - **menuId:** The ID of the menu displayed in the view. The default is `CoordinateActions`.

## View Models

- **PlotCoordinatesViewModel:**
  - **isEnabled:** A boolean that sets whether the view model is enabled. The default value is `true`.
  - **coordinatesModel:** Sets the coordinates view model. The default value is `MapCoordinatesModel`, which is a view model configured as part of the [Map Module](#).
- **CoordinateActionsViewModel:** None

## See also...

[CompactToolbar Module on page 150](#)

[Snapping Module](#) on page 360

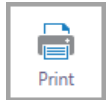
[TabbedToolbar Module](#) on page 363

## 15.55 Printing Module

The Printing Module implements simple printing of the map image.

In order to print a map, a print template must be installed in Manager. You can create print templates in Geocortex Report Designer. Refer to the Report Designer help system for information.

A factory HTML5 Viewer has a tool in the toolbar to print the map. If the site does not have any print templates, the Print tool is grayed out in the toolbar.



### Print tool

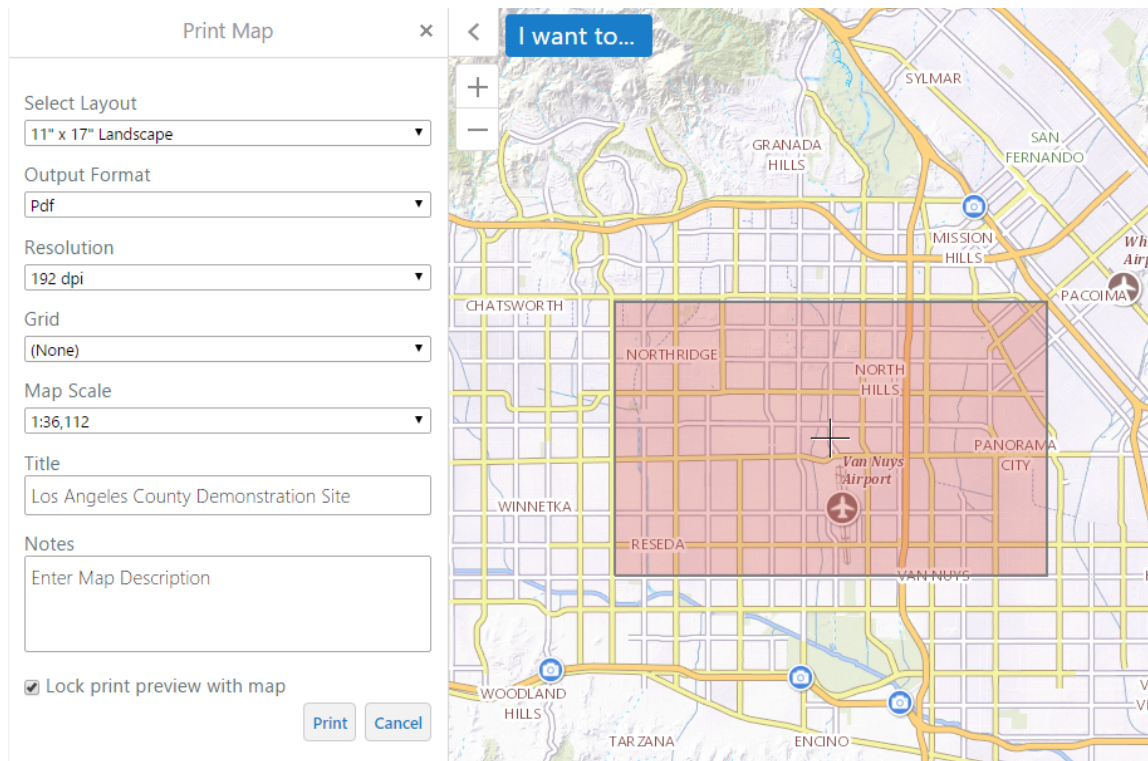
You can edit the template in Manager to configure options that allow the user to select the map scale and resolution, as well as textual content such as the title or notes to add to the printout. For information on configuring print templates, refer to the *Geocortex Essentials Administrator Guide*.

## Print Preview

### Desktop and Tablet Interfaces

In the Desktop and Tablet interfaces, the map previews the extent to be printed using an overlay. The user can adjust the print preview overlay to match their printout exactly. The procedure to adjust the extent is as follows:

1. Pan and zoom the map to center the area of interest in the map viewport.  
Zooming the map does not change the scale at which the map prints.
2. Select the **Lock print preview with map** checkbox to lock the print preview overlay to a position on the map.  
Deselect the checkbox to move the print preview extent around the map.
3. Change the **Map Scale** setting as needed to ensure the overlay covers the desired area.  
The Map Scale setting's Current Extent option guarantees that everything in the viewport will be printed.



### Extent preview shown in the Desktop interface

The user clicks **Print** and waits to be prompted to download the printout.

### Handheld Interface

In the Handheld interface, clicking the Print tool opens the Print Map panel in the Bottom Panel. The print preview overlay is not shown initially.



The Handheld interface does not use `UnlockedPrintPreviewView`. Instead, users can look at the print preview extent with the **Center print here** and **Hide print preview** buttons in the Print Map panel.

To adjust the print preview extent, the user follows this procedure:

1. Pan and zoom the map to center the area of interest in the map viewport.  
Zooming the map does not change the scale at which the map prints. The user may need to zoom out to see the extent of the overlay.
2. Change the **Map Scale** setting as needed to ensure the print preview overlay covers the desired area.  
The Map Scale setting's Current Extent option guarantees that everything in the viewport will be printed.
3. Repeat steps 1 and 2 until the map viewport displays the area you want to print at the desired scale.
4. Clicks **Center print here**.  
The print preview overlay displays on the map and the map zooms out so the entire overlay is visible. This does not change the scale used for the printout—the Viewer uses the scale selected when the print was centered.

The user clicks **Print** and waits to be prompted to download the printout.

## Print Output Settings

The Print Map panel allows users to configure how their extent will be printed.

- **Select Layout:** Lists the site's available print templates.
- **Output Format:** Lists the available file formats to send the printout to. (For example, a PDF.)
- **Resolution:** Lists the available resolutions for the printout in DPI (dots per inch).
- **Grid:** Configures whether a grid overlay measures the map printout in latitude/longitude or map units, or, alternatively, whether the grid overlay is hidden.

The selected print template may add additional fields to the Print Output settings. This depends on whether a print template has other fillable fields configured, such as a title or description.

## Configuration Properties

### Module

None

### Views

- **PrintingView:** None
- **UnlockedPrintPreviewView:** None



The Handheld interface does not use `UnlockedPrintPreviewView`. Instead, users can look at the print preview extent with the **Center print here** and **Hide print preview** buttons in the Print Map panel.

### View Models

- **PrintingViewModel:**
  - **previewBackgroundColor:** The background color for the print preview extent in an RGBA color format (i.e. [211, 211, 211, 0.6]).
  - **previewBorderColor:** The print preview extent overlay border in an RGB color format (i.e. [0, 0, 255]).
  - **previewBorderThickness:** An integer that sets the print preview extent overlay border thickness in pixels.
  - **previewCenterImageUrl:** A relative URL to the image used to mark the center of the print preview extent overlay.

## 15.56 Project Module

Projects allow end users to save the state of the viewer, share the saved viewer state, and reload it later.

The ability to share projects makes them a valuable collaboration tool. For example, a user could open the viewer, pan to a location, add markup, and then save the project. The user can then share the project with colleagues so they can load the project, see the user's changes, and add their own changes.

Projects belong to the site of the viewer that created the project. A user cannot create a project in one viewer and then load the project in a viewer that belongs to a different site. If the site has more than one HTML5 viewer, a project can be created in one HTML5 viewer and loaded in the site's other HTML5 viewers.



HTML5 Viewer projects require Geocortex Essentials version 4.5 or newer.

## Projects and the Data Store

Projects are saved on the server, in the Geocortex Data Store. Geocortex Essentials Manager provides an interface to the Data Store that administrators can use to manage projects. For information, see "Geocortex Data Store" in the *Geocortex Essentials Administrator Guide*.

## What Projects Save

When you save a project, the following aspects of the viewer's state are saved:

- Map's center point
- Currently selected basemaps
- Layer visibility
- Currently applied layer filters
- Currently selected layer theme
- User-added layers, including layers added dynamically from a layer catalog
- Uploaded files such as CSV files or shapefiles
- Markup (graphics and text) from measurement and drawing tools
- List of saved filters and queries
- Current feature set, and where it is displayed (Results List or Results Table)
- Current set of selected features
- Current feature details, and how they are displayed (compact or expanded)
- Heat maps
- Clustering
- Highlights
- Pushpins
- Plotted coordinates
- Current map tip
- Current mouse coordinates selection
- Bookmarks

## Map Center Point

Instead of saving the map extent, the project saves the map's center point. This preserves the scale level when the project is loaded into a map viewport that has different dimensions than the original viewport.

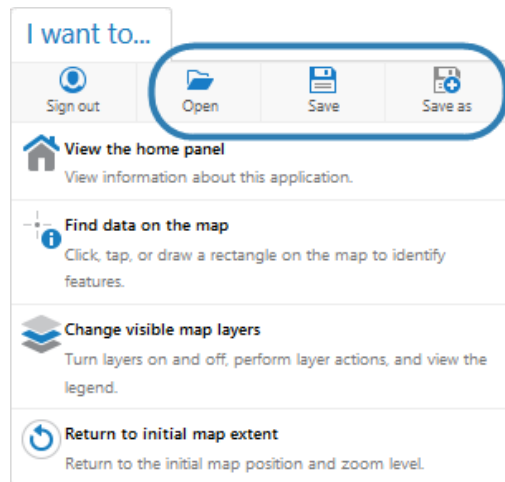
## Project Bookmarks

A bookmark is not restored from a project if a bookmark with the same name already exists in the user's personal bookmarks. This ensures that a user's personal bookmarks are not overwritten by project bookmarks.

Project bookmarks are not saved with the user's personal bookmarks. Project bookmarks are available only while the project is loaded.

## Project Tools

By default, the I Want To menu's [Global menu](#) has tools for working with projects, shown in the image below.



**Open** - Open the Projects panel. The Projects panel lists the projects that this user is allowed to load. The user clicks a project in the list to load the project. This restores the viewer state that is saved in the project.


**Save** - If a project is currently loaded and the user has the necessary privileges, update the project with the current viewer state. Otherwise, create a new project using the name and description entered by the user.

**Save as** - Create a new project using the name and description entered by the user.

### Project tools, shown in the Desktop interface

You can also add the project tools to the Tabbed Toolbar. In the list of available tools, you can find Open, Save, and Save as tools in the **Global Tasks** tool group. If you configure the viewer to use the Full GIS Toolbar, then users can access the project tools on the toolbar's File tab. For more information about toolbar configuration, see the "Configure the Toolbar" topic in the *Geocortex Essentials Administrator Guide*.


## Creating Projects

 The user must be signed in to the viewer to create a project.

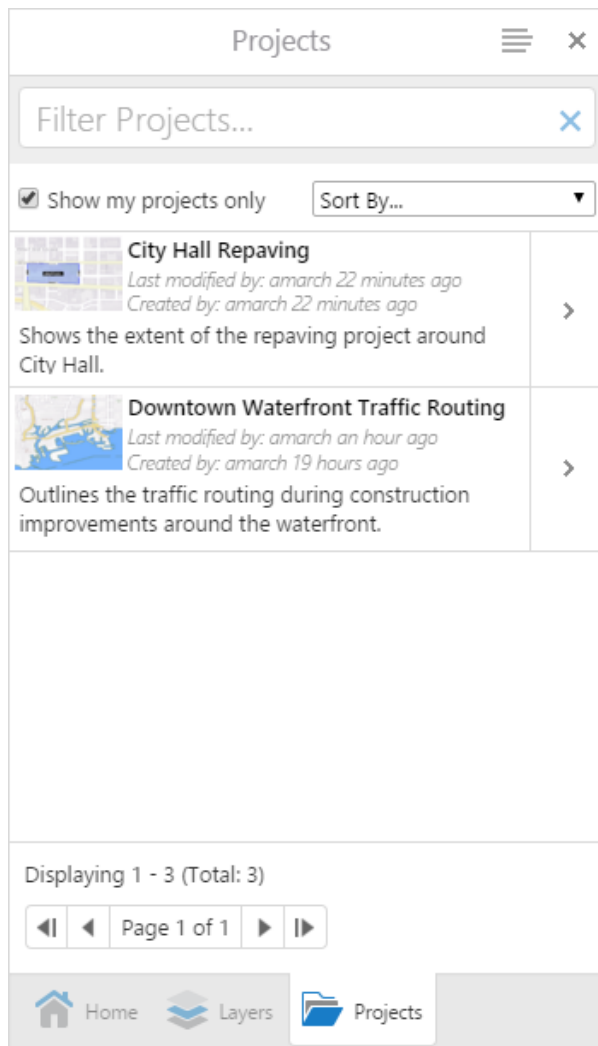
To create a project, the user opens the I Want To menu and selects **Save**. The Save Project panel appears, where the user enters a unique name and optional description, and then chooses **Save** to create the project.

Alternatively, the user can click **Save as** instead of Save, provided a project is not currently loaded. If a project is already loaded, the Save function saves the loaded project instead of creating a new one.

## Loading Projects

 The user does not need to be signed in to load a project. However, this requires additional configuration in Manager. For information about allowing access to **Everyone (no sign-in required)**, see [Access Privileges](#) and [Guest Links](#).

To load a project, the user opens the I Want To menu and selects **Open** to open the Projects panel. The user can then select a project from the projects list.



Projects panel, showing two projects

## Project Ownership

To create a project, the user must be signed in to the viewer. Users who access the viewer under Anonymous Access cannot create projects.

The user who creates a project is the project's **owner**. Project owners have full access to the project:



- **Share Project:** By default, a project can be seen only by the project's owner. The project's owner can share the project with other users, so that they can see the project on the Projects panel and load the project.
- **Update Project:** The project's owner can update the viewer state that is saved in the project.
- **Update Project Details:** The project's owner can change the project details, such as the project name.
- **Delete Project:** The project's owner can delete the project.

You can change a project's owner in Geocortex Essentials Manager. For instructions, see "Geocortex Data Store" in the *Geocortex Essentials Administrator Guide*.

## Sharing Projects

By default, only the project owner can see the project. It does not appear in the Projects panel for other users. The project owner can share the project with other users from the Share Project panel.

The project owner can open the Share Project panel at any time by following the steps below:

1. Open the I Want To menu, and select **Open**.
2. Use the  Action icon beside the project that you want to share.
3. Select  **Share**.

The Share Project panel opens.



Share Project ✕

**Project Privileges**  
Project privileges allow the users that you specify to access the project using the Project URL or by browsing the list of projects.

Project URL

Share with

	None	View	Edit
Everyone (sign-in required)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

**Guest Links**  
Guest links allow users who are not assigned project privileges to access the project.

View Link (sign-in not required)

Edit Link (sign-in required)


Home
 Layers
 Share Project

**Default appearance of the Share Project panel**

Other users can also open a project's Share Project panel using the steps above. However, unless they are the project owner, the access privileges and guest links are not visible.

The method that the owner uses to share a project with other users depends on whether the users will sign in to the viewer or not:

- **Access Privileges:** Share projects with users who can sign in to the viewer. To share the project, the owner sends the Project URL to the users, or the users browse for the project in the viewer.
- **Guest Links:** Share projects with users who cannot not sign in to the viewer (for example, temporary contractors that have not been added to your security provider). To share the project, the owner sends a guest link to the users.

 Access privileges and guest links do not allow users to bypass site security. Site security is enforced no matter what method is used to share a project.


 Configuring access privileges does not have any effect on the permissions that are configured in the site.

## Access Privileges

The project's owner can assign access privileges to other users and groups of users. For example, if the project owner signs in to the viewer using ArcGIS Online credentials, then the owner can assign access privileges to ArcGIS Online users and groups. Similarly, if the owner signs in using Identity Server, then the owner can assign privileges to Identity Server users.

The owner can assign access privileges to:

- **Everyone (sign-in not required):** Share the project with everyone. This allows anyone who can access the viewer, including users who access the viewer using Anonymous Access, to load the project. Users do not have to sign in—the project is public.

 This setting is only available in the viewer if the Geocortex Data Store is configured to Allow Public Read Access. For information, see "Geocortex Data Store" in the *Geocortex Essentials Administrator Guide*.

- **Everyone (sign-in required):** Share the project with all signed-in users. This allows anyone who can sign in to the viewer using any of the enabled security providers to access the project. Anonymous users cannot see or load the project.

For example, suppose you have configured the site to allow sign-in using ArcGIS Online or Windows credentials, and you also allow Anonymous Access. A project's owner gives **Everyone (sign-in required)** the **Edit** privilege. Any user who can sign in to the viewer using ArcGIS Online or Windows credentials can load and update the project. Anonymous users cannot see or load the project.

- **Groups:** Share the project with specific groups of users.

The project's owner can search for groups to assign access privileges to. Searching for groups searches the security provider that the project's owner signed in with. For example, if the owner signs in with Windows Authentication, Active Directory groups are searched.

To remove a group from the list, the project's owner selects the **None** option for that group and chooses **Save Sharing**.

- **Individuals:** Share the project with individual users.

The project's owner can search for individuals. Searching for individual users searches the security provider that the project's owner signed in with. For example, if the owner signs in with Portal for ArcGIS, Portal for ArcGIS users are searched.

To remove an individual user from the list, the project's owner selects the **None** option for that user and chooses **Save Sharing**.

The access privileges are:

- **None:** Users cannot see the project in the Projects panel and they cannot load the project. If the project's owner selects None for a specific group or user, the group or user is crossed out in the Share Project panel. When the owner chooses **Save Sharing**, the group or user is removed from the list. Selecting the None option does not remove Everyone (sign-in not required) or Everyone (sign-in required) from the list.



The View privilege takes precedence over None, and the Edit privilege takes precedence over the View privilege. If more than one access privilege applies to a particular user or group, the highest privilege applies. For example, if a particular user has None privilege and they belong to a group that has Edit privilege, the user will be able to edit the project.

- **View:** Users with the View privilege can see the project in the Projects panel and load the project. They can also open the Share Project panel, but they can only see the Project URL. The access privileges and guest links are hidden, and they cannot update the project details or delete the project.
- **Edit:** Users with the Edit privilege can see the project in the Projects panel, load the project, and update the project. They can also open the Share Project panel, but they can only see the Project URL. The access privileges and guest links are hidden, and they cannot update the project details or delete the project.

## Guest Links

A project's owner can create guest links (URLs) that enable other users to launch the viewer and load the project. There are two types of guest link:


- **View Link:** The View Link allows users to load the project. Users cannot update the project, update the project details, or delete the project. If the Data Store policy is **Deny Public Read Access**, users must sign in to load the project. The other policies allow users to load the project using a View link without signing in. For more information, see [Effect of Data Store Policies on Projects on page 312](#).
- **Edit Link:** The Edit Link allows users to load the project and also update the project. Users cannot change the project details or delete the project. If the Data Store policy is **Allow Public Read Access and Public Edit Links**, users do not have to sign in to edit the project. The other policies require users to sign in to edit the project. For more information, see [Effect of Data Store Policies on Projects on page 312](#).





To share guest links with users who are not signed in, you need to configure the Geocortex Data Store to Allow Public Read Access. For information, see "Geocortex Data Store" in the *Geocortex Essentials Administrator Guide*.



We recommend that you only share guest links, especially the Edit guest link, when you have no other option.

To create a guest link, the project's owner clicks the  **Add Link** tool for the desired type of guest link and then clicks **Save Sharing**. The owner then sends the link (URL) to the user. Once a project owner has shared a guest link with a user, the user can share the link with other people.

The project's owner can revoke guest links so the links no longer work. To revoke a guest link, the owner clicks the  **Remove Link** tool for the desired type of link, and then clicks **Save Sharing**. This removes the guest links from the Share Project panel and stops them from working. After the owner removes a guest link, users can no longer use the guest link to launch the viewer or load the project.

Removing a guest link does not stop user sessions that are already in progress. However, if the user used an Edit link, they lose the ability to update the project as soon as the owner removes the link. The owner can create a new View link or Edit link using the  **Add Link** tool.



Once a project owner has removed a guest link and clicked Save Sharing, there is no way to make the link work again. Clicking the Add Link tool does not restore the old guest link—it creates an entirely new link.

## Effect of Data Store Policies on Projects

The Data Store policy that is configured in Manager determines whether the guest links require users to sign in. The following table summarizes how the different policies affect sign-in using guest links.

### Effect of Data Store Policy on Guest Link Sign-in

Data Store Policy	View Links	Edit Links
Deny Public Read Access	Requires sign-in	Requires sign-in
Allow Public Read Access	Does not require sign-in	Requires sign-in
Allow Public Read Access and Public Edit Links	Does not require sign-in	Does not require sign-in



If you change the Data Store policy, existing links will behave according to the new policy, not the policy that was in effect when the link was created. For example, if you change the policy from Deny Public Read Access to Allow Public Read Access, existing View links will no longer require users to sign in.

For more information, see "Configure Data Store Policies" in the *Geocortex Essentials Administrator Guide*.

## Sharing Panel Summary

The screenshot shows the 'Share Project' dialog box with the following sections and annotations:

- Project Privileges:**
  - Project URL:** A text field containing a URL. Annotation: "URL to share with users who will sign in to the viewer".
  - Share with:** A dropdown menu set to 'Individuals' and a search field. Annotation: "Search for groups or individuals to assign access privileges to".
  - Add:** A button to add new entries to the list.
  - Access Privileges Table:**

	None	View	Edit
Everyone (sign-in not required)	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
Everyone (sign-in required)	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
amarch	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

 Annotation: "List of access privileges assigned to groups and individuals. Only the project's owner can assign access privileges. One individual has been added to the list (amarch). The options that show in the shaded area depend on the Data Store policy that is configured in Manager."
- Guest Links:**
  - View Link (sign-in not required):** A text field with a '+' button. Annotation: "Create links to share with users who will not sign in. The View Link does not allow users to update the project. The Edit Link allows users to update the project. Only the project's owner can create guest links."
  - Edit Link (sign-in required):** A text field with a '+' button. Annotation: "If you don't want to allow guest links, you can hide the Guest Links area from project owners."
- Buttons:**
  - Stop Sharing:** Annotation: "Delete all the access privileges and revoke the guest links".
  - Save Sharing:** Annotation: "Save the access privileges and guest links".
  - Cancel:** A button to close the dialog.

Summary of Share Project panel options for the project's owner

## Example: Use a Project to Collaborate

Taylor and Bill are working on development plans for an industrial site that their company is developing. To start the planning, Bill marks up a site map with notes and ideas. First he signs in to the viewer, searches for the parcel, and zooms the map to it. Then he draws some areas of interest, which he annotates using the Text tool. He also makes some preliminary measurements using the measurement tools. When he is satisfied that he has done everything he wants, he creates a project (**I Want To | Save as**), which he names Site Plan. In the confirmation dialog box, he clicks **Share** to open the Share Project panel.

To share the project with Taylor, Bill searches for Taylor's name and adds it to the list of users and groups. He sets Taylor's access privilege to **Edit** so Taylor can make changes and update the project. Bill also searches for his assistant, Marcus, and gives him **View** privilege. Finally, Bill copies the **Project URL** and emails it to Taylor and Marcus. He clicks **Save Sharing** and closes the Share Project panel.

At this point, Bill realizes that he has forgotten a particular measurement. He selects the measurement tool, adds the missing measurement, and updates the project (**I Want To | Save**). When his colleagues open the project, they will see the most recent version of the project, which contains Bill's final measurement.

When Taylor receives the email with the project URL, she copies the URL to her browser's address bar and presses Enter. She signs in to the viewer when prompted and the project loads. The map and the saved markup load. Taylor reviews Bill's notes and markup. After reviewing Bill's work, Taylor edits some of Bill's markup, adds some new markup, and saves the project. She then sends Bill and Marcus an email telling them that she has updated the project.

Marcus signs in to the viewer and opens the Projects panel (**I Want To | Open**). He clears the **Show my projects only** filter so he can see projects created by other users for which he has access. He sees a project called Site Plan and opens the project. Because Taylor has already updated the project, Marcus sees the current version of the project, with Taylor's changes. He reviews what Bill and Taylor have done and adds some markup of his own to the map, but he cannot update the project because he only has View privilege. To keep a copy with his markup, he clicks **Save** and creates a new project.

When Bill receives Taylor's email, he reloads the project to review Taylor's markup. Bill and Taylor continue to collaborate on the site plan by marking up the map and updating the project. Marcus stays up to date on the planning by viewing the project after changes have been made.

## Project Commands and Events

The Project Module provides commands and events that you can use to work with projects programmatically. The commands and their related events are listed below. All the project commands can be used in hyperlinks, menus, toolbar tools, and workflows. For more information about commands and events, refer to the HTML5 Viewer's API Reference. For instructions on opening the API Reference, see [Geocortex SDK for HTML5 API Reference on page 422](#).

- **SaveAsProject:** Displays the user interface to fill in the details about a new project. The viewer raises a `ProjectSavingEvent` when saving begins, and a `ProjectSavedEvent` when saving completes successfully. If saving fails, the viewer raises a `ProjectErrorEvent`.
- **SaveProject:** Updates the viewer state in the currently loaded project, or, if a project is not currently loaded, displays the user interface to fill in details about the new project. The viewer raises a `ProjectSavingEvent` when saving begins, and a `ProjectSavedEvent` when saving completes successfully. If saving fails, the viewer raises a `ProjectErrorEvent`.
- **ShowShareProject:** Displays the user interface for sharing the specified project.
- **ShowProjects:** Displays the user interface for browsing and loading projects.
- **RefreshProjectsList:** Refreshes the list of projects that the user can browse and load.
- **LoadProject:** Loads the specified project in the viewer. The viewer raises a `ProjectLoadingEvent` when loading begins, and a `ProjectLoadedEvent` when loading completes successfully. If loading fails, the viewer raises a `ProjectErrorEvent`.
- **ShowProjectEditor:** Shows the user interface for editing the details for the specified project. The viewer raises a `ProjectEditorFinishedEvent` when editing completes.

- **DeleteProject:** Deletes the specified project. The viewer raises a `ProjectDeletingEvent` when deletion begins, and a `ProjectDeletedEvent` when the project is successfully deleted. If deletion fails, the viewer raises a `ProjectErrorEvent`.

## Configuration Properties

### Module

None

### Views

- **ProjectActionsView:** This view shows options for managing and sharing projects.
  - **menuId:** The ID of the menu to show in this view. The default is `ProjectActions`. By default, `ProjectActions` has three menu items: **Share**, **Edit Details**, and **Delete**. The menu itself is configured in the [Menu Module](#).
- **ProjectEditorView:** This view shows the interface for editing a project's details.

None
- **ProjectStatusView:** This view shows a status indicator when a project is loading or being saved.

None
- **ProjectsView:** This view shows the Projects panel, which lists the projects that the user has access to.

None
- **ShareProjectView:** This view shows the Share Project panel, which allows the project's owner to assign access privileges and create guest links.

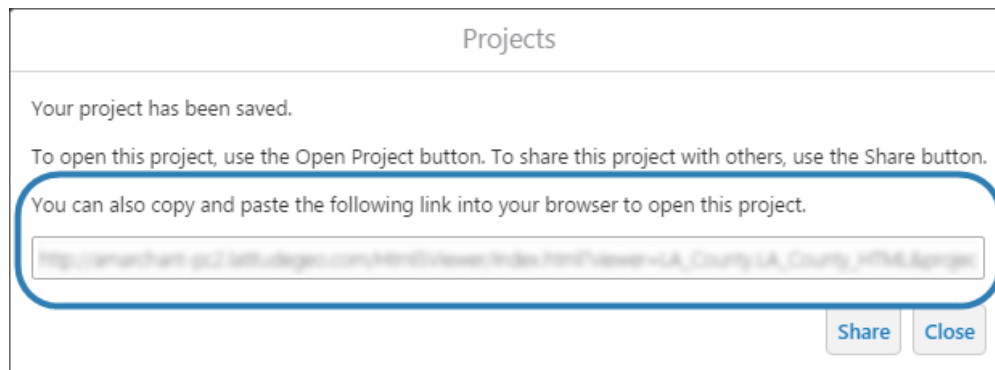
None

### View Models

- **ProjectActionsViewModel:** This view model is used by `ProjectActionsView`.

None
- **ProjectEditorViewModel:** This view model is used by `ProjectEditorView`.

None
- **ProjectStatusViewModel:** This view model is used by `ProjectStatusView`.
  - **showUrlOnSave:** When this property is `true`, the project's URL shows in the confirmation dialog that displays when a project is saved. If you do not want the confirmation dialog to show the project's URL, set `showUrlOnSave` to `false`.



Confirmation dialog showing the content that is controlled by the `showUrlOnSave` property

- **ProjectsViewModel:** This view model is used by `ProjectsView`.
  - **defaultThumbnail:** The thumbnail image to show for this project on the Projects panel if the project's thumbnail fails. The default is `Resources/Images/Icons/map-no-preview-70x50.png`.
  - **minimumFilterDelay:** The number of milliseconds to wait after the user stops typing in the Filter Projects box before starting to filter. The default is `300`. The Filter Projects box is on the Projects panel.
- **ShareProjectViewModel:** This view model is used by `ShareProjectView`.
  - **confirmSharedPublicly:** When this property is `true`, a confirmation prompt displays when the user sets **Everyone (sign-in not required)** to **View** or enables a guest link that does not require sign-in, and clicks **Save**. If you do not want the viewer to prompt for confirmation, set `confirmProjectViewModel` to `false`. The default is `true`.
  - **showGuestLinks:** When this property is `true`, the Guest Links area shows on the Share Project panel for project owners. If you do not want the Guest Links area to show, set `showGuestLinks` to `false`.

#### See also...

[TabbedToolbar Module on page 363](#)

[GlobalMenu Module on page 187](#)

## 15.57 Prompt Module

The Prompt Module creates and displays customized prompt dialogs.

### Configuration Properties

#### Module

- **promptRegion:** The region where the prompt is displayed. The default is `ModalWindowRegion`.

#### Views

- **PromptView:** None



View Models

None

## 15.58 Pushpins Module



Pushpins are not currently supported in the Handheld interface. Adding pushpin configuration to the Handheld configuration file has no effect.

The Pushpins Module is responsible for placing a pushpin on the map. When pushpins are visible, they are placed in the center of a feature, subject to any configured offset. Pushpins can be used with geocoders and workflows.

For information about pushpin visibility for point, line, and polygon features on a map, see [Configure Pushpins on page 104](#).

## Configuration Properties

## Module

- **pushpinsEnabled**: If you want pushpins to appear for each feature of the search results, set to `true`; otherwise, set to `false`. The default is `true`.
- **pushpinsRemainVisible**: If you want pushpins to remain visible even when a view other than the Results List or Results Table activates, set to `true`; otherwise, set to `false`. Closing the Results panel will still hide the pushpins regardless of this property. The default is `true`.



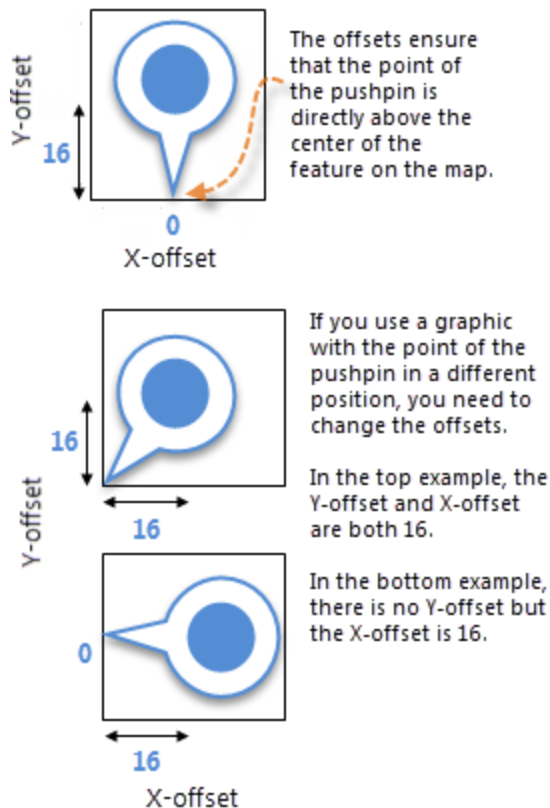
In versions prior to 2.6, the default behavior was to hide pushpins when a view other than the Results List or Results Table activates. If you want the previous default behavior, set this property to `false`.



This option has been deprecated effective with Essentials Manager 4.7 as a result of changes implemented in highlighting. For further information about highlighting, see [Highlight Module on page 191](#).

- **pushpinMarkerUri**: The URL to the image that represents the pushpin. The default is a red pushpin image located at `Resources/Images/Pushpins/map-marker-red-32.png`. Other colors available out-of-the-box include blue, green, purple and yellow.
- **pushpinMarkerWidth**: The width of the pushpin in pixels. This is typically set to the width of the pushpin image, otherwise the image will be scaled. The default is 32.
- **pushpinMarkerHeight**: The height of the pushpin in pixels. This is typically set to the height of the pushpin image, otherwise the image will be scaled. The default is 32.
- **offsetX**: The number of pixels to horizontally offset the pushpin image. In other words, the distance along the X-axis between the center of the feature and the center of the pushpin image. Use a negative integer to place the pushpin to the left of the feature's center. The default is 0.


- **offsetY**: The number of pixels to vertically offset the pushpin image. In other words, the distance along the Y-axis between the center of the feature and the center of the pushpin image. Use a negative integer to place the pushpin below the feature's center. The default is 16, because the pointer is at the bottom of the default image, which is 32 pixels high.





- **behaviors**: An array of named behaviors that run when an associated event occurs. By default, there is a single behavior, `PushpinClickedBehavior`, although several other behaviors exist that are omitted by default:
  - **PushpinClickedBehavior**: A behavior that runs an array of commands when the user clicks a pushpin. By default, this includes two commands: `ShowMapTip` and `HighlightFeatureDefault`.
  - **PushpinMouseDownBehavior**: A behavior that runs an array of commands when the left mouse button is pressed down while the mouse pointer is on a pushpin. By default, this behavior is omitted.
  - **PushpinMouseRightButtonDownBehavior**: A behavior that runs an array of commands when the right mouse button is pressed down while the mouse pointer is on a pushpin. By default, this behavior is omitted.
  - **PushpinMouseLeftButtonUpBehavior**: A behavior that runs an array of commands when the left mouse button is released while the mouse pointer is on a pushpin. By default, this behavior is omitted.
  - **PushpinMouseRightButtonUpBehavior**: A behavior that runs an array of commands when the right mouse button is released while the mouse pointer is on a pushpin. By default, this behavior is omitted.
  - **PushpinMouseEnterBehavior**: A behavior that runs an array of commands when the mouse pointer

first hovers over a pushpin. By default, this behavior is omitted.

- **PushpinMouseLeaveBehavior**: A behavior that runs an array of commands when the mouse pointer moves away from a pushpin it was previously hovering over. By default, this behavior is omitted.

 You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.

 You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.

 Adding a new behavior is only recommended for experienced developers.

## Views

None

## View Models

None

## Example: Add a Command with a Parameter to a Behavior

The following example demonstrates adding a new command with a parameter to the behavior, `MapOnFeatureClickBehavior`.

### To add a command with a parameter to a behavior:

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Sites\[site]\Viewers\
[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. In the `Pushpins` module section, find the `behaviors` property. Locate the behavior you want to edit. For example, `PushpinClickedBehavior`.

```
{
  "moduleName": "Pushpins",
  ...
  "configuration": {
    ...
    "behaviors": [
      {
        "name": "PushpinClickedBehavior",
        "commands": [
          "ShowMapTip",
          "HighlightFeatureDefault"
        ]
      }
    ]
  }
}
```

The behavior executes two commands: **ShowMapTip** and **HighlightFeatureDefault**.

4. Consult the Geocortex SDK for HTML5 API Reference to determine the type of parameter that is associated with these commands. Both commands a parameter of type, `geocortex.essentialsHtmlViewer.mapping.infrastructure.Feature`.
5. Find a command that you want to add to the behavior in the Geocortex SDK for HTML5 API Reference that has the same parameter type. For example, **PanToFeature**.
6. Add the desired command to the list of commands, separated by a comma. For example:

```
"commands": [
  "ShowMapTip",
  "HighlightFeatureDefault",
  "PanToFeature"
]
```

7. Save the file.

#### See Also...

[MapTips Module on page 253](#)

## 15.59 QueryBuilder Module

The QueryBuilder Module implements the Query Builder and Filter Builder.

## Query Builder



WMS layers that are not associated with a WFS do not support query operations.

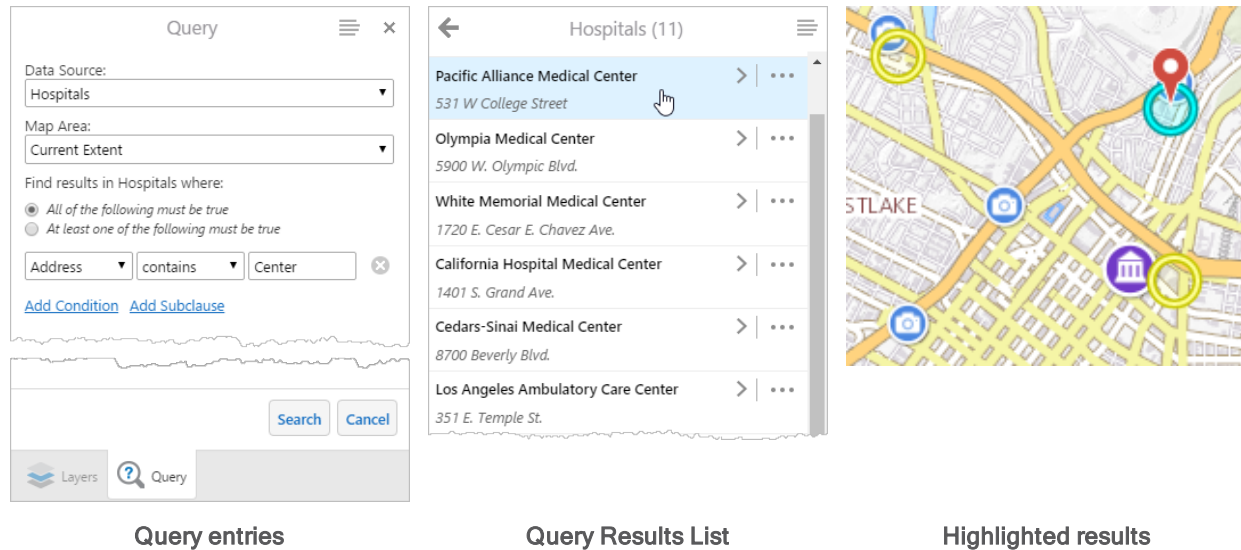
The Query Builder enables end users to construct queries that search for specific features on a layer.

### Query panel - fields and options

1	Data Source: lists the feature layers. Select one from the list.
2	Map Area: indicates the scope for the query. Select <b>All</b> or <b>Current Extent</b> .
3	Find results area: Indicates the selected data source name that the query applies to. Select <b>All of the following must be true</b> to use the AND operator in the query. Select <b>At least one of the following must be true</b> to use the OR operator in the query.
4	Expression fields: Select the query subject in the field on the left. Select the operator for the expression in the center field (the list of operators is dependent on whether the subject represents an alphanumeric or numeric value). Type the value being searched for in the field on the right.
5	Add Condition: If <b>All of the following must be true</b> (the AND operator) is selected, AND expression fields are added to the query. If <b>At least one of the following must be true</b> (the OR operator) is selected, OR expression fields are added to the query.
6	Add Subclause: Adds a nested clause to the expression. The subclause allows the user to nest an AND or OR expression by selecting the respective option. Note that the user can nest only one subclause expression. (Queries created by an administrator in Manager can contain nested subclauses. See the "Queries" topic in the <i>Geocortex Essentials Administrator Guide</i> .)
7	Search: Click <b>Search</b> to run the query.

The results of a query are listed in the **Query Results** panel, and each item is highlighted on the map. The features do not need to be visible at the current map scale for highlighting to be shown.

For example, a user can define a query to search the **Hospitals** layer in the current map extent for medical facilities containing **Center** in the name. Clicking **Search** lists those medical facilities in the **Query Results** panel, and highlights each one on the map with yellow circles. If you place focus on a specific item in the list, the highlight color for that item changes to blue on the map. (See [Highlight Module on page 191](#) for information about highlighting and emphasizing features.)



The user can click a medical center in the query results list to see additional details about it. (If the feature is visible on the map, they can also click it to open a map tip and click the **View Additional Details** link.)

To save the query, click the **Query** tab at the bottom of the **Query Results** panel, and click **Save Query As** in the Panel Actions menu. If the user changes a query that was previously saved, they click **Save Query**. Note that user-defined queries that are saved are available only in the current Viewer session. To save a query for use in a later session, the user needs to save the project.

Queries can be created by administrators in Manager. Administrator queries are included in the **Saved Queries** panel as *Administrator Defined* queries, which are read only and they can be run by any user. (See the "Queries" topic in the *Geocortex Essentials Administrator Guide* for information about creating administrator queries.) Users can edit administrator queries and use **Save Query As** to save a user-defined copy of the administrator query, but they cannot change the administrator query itself.

## Filter Builder

The Filter Builder enables end users to filter the features that show on the map. Filtering does not affect which features are listed in search or query results. Filtering is the same as applying a definition expression in ArcMap.

The Filter panel contains the same fields and options as the Query panel, but it includes **Clear** and **Filter** buttons at the bottom rather than **Search**. (See [Query Builder on page 321](#) for information about the fields and options.) Clicking **Filter** shows the features specified in the **Filter** panel and hides those that do not match the filter criteria. Clicking **Clear** shows the features that were hidden by filtering.

For example, a user could filter the **Hospitals** layer for medical facilities with **Hospital** in the name that provide **Emergency Services** and are open **24 Hours**. Clicking **Filter** hides medical facilities that are not hospitals with 24-hour emergency services. Clicking **Clear** re-displays the hidden features on the map.

## Configure the Query Builder and Filter Builder



User added feature layers and dynamic layers can be queried and filtered without configuration.

To enable end users to use the Query Builder and Filter Builder, you must:

- **Enable Querying:** Enable querying on the layers that you want users to be able to query and filter. The **Allow Query Operations** setting controls whether a layer can be queried and filtered. You can configure the Allow Query Operations setting for a layer by editing the layer, by using the layer's drop-down menu, or by using a batch editing menu. For instructions, see "Layer Settings" in the *Geocortex Essentials Administrator Guide*.
- **Configure the Data Provider:** Depending on the data provider that you use to store feature data, you might have to configure the data provider in Manager. In some cases, you might also have to configure some properties related to the data provider in the viewer's configuration files. Configuring the data provider enables the Query Builder and Filter Builder to construct queries using the correct syntax for your data provider. For instructions on configuring the data provider, see "Map Service Functional Tab" in the *Geocortex Essentials Administrator Guide*.
- **Provide Access:** You must provide a way for end users to access the Query Builder and Filter Builder.

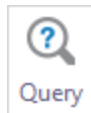
- **Query Builder:** To provide access to the Query Builder for end users:

- Create a hyperlink, I Want To menu item, or workflow that uses the `ActivateView` command to activate the Query Builder's view:

```
ActivateView(SimpleQueryBuilderView)
```

- Add the **Query** tool to the toolbar.

For instructions on adding tools to the toolbar, see [Configure the Toolbar on page 108](#).



Query tool

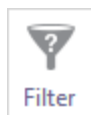
- **Filter Builder:** To provide access to the Filter Builder for end users:

- Create a hyperlink, I Want To menu item, or workflow that uses the `ActivateView` command to activate the Filter Builder's view:

```
ActivateView(SimpleFilterBuilderView)
```

- Add the **Filter** tool to the toolbar.

For instructions on adding a tool to the toolbar, see [Configure the Toolbar on page 108](#).



Filter tool

## Configuration Properties



If your data provider uses the ANSI SQL default standard or one of the well-known data providers that are listed in the **(Default) Data Provider** drop-down list in Manager, you do not need to configure these properties. You only need to configure these properties if your data provider uses some other query syntax.

The HTML5 Viewer supports the use of the following query tokens in Query Builder and Filter Builder format strings:

- **Field Name:** {0} is substituted at run time for the name of a field in the layer that is being queried or filtered.
- **Input Value:** {1} is substituted at run time for the input value entered by the user. All input is text.
- **Date Value:** {0:[date format]} is substituted at run time for a date value. Replace [date format] with the date format that your data provider uses.

### Module

- **queryStatusRegion:** The region in the **Query Results** list used for query status messages. The default is `QueryBuilderStatusRegion` for Desktop and Tablet interfaces, and `ResultsStatusRegion` for the Handheld interface.
- **filterStatusRegion:** The region in the **Filter** panel used for filter status messages. The default is `FilterBuilderStatusRegion` for Desktop and Tablet interfaces, and `ResultsStatusRegion` for the Handheld interface.

### Views

- **SimpleQueryBuilderView:** This view is used when the end user selects the Query Builder.
  - **wildcard:** The character that your data provider uses as a wildcard. For example, the ANSI SQL default wildcard is %. You cannot use query tokens in this format string.
  - **dateQueryFormat:** The date format that your data provider uses. The syntax of the date format is {0:[date format]}, where 0 is the field name and [date format] is the date format that your data provider uses. Query Builder uses the `dateQueryFormat` property to construct date queries. Your data provider's date format can contain any of the [.NET date and time format specifiers](#) that are listed below. For example, the date format for the ANSI SQL default data provider is `DATE: '{0:yyyy-MM-dd}'`. This uses three .NET format specifiers: `yyyy`, `MM`, and `dd`.

d	ffffff	h	s
dd	F	hh	ss
f	FF	H	y
ff	FFF	HH	yy
fff	FFFF	m	yyy
ffff	FFFFF	mm	yyyy
fffff	FFFFFF	M	yyyyy
ffffff	FFFFFFF	MM	

- **textComparisonQueryFormat:** The format of a query that compares the value of a text field to the



- input value. For example, the format for text comparisons in the ANSI SQL default data provider is `LOWER({0}) LIKE LOWER({1})`.
- **numberToTextComparisonQueryFormat**: The format of a query that compares the value of a numeric field to the input value. For example, the format for number-to-text comparisons in the ANSI SQL default data provider is `CAST({0} AS VARCHAR(50)) LIKE '{1}'`.
  - **doesNotContainQueryFormat**: The format of a query that tests whether a field contains the input value. For example, the format for text comparisons in the ANSI SQL default data provider is `LOWER({0}) NOT LIKE LOWER({1})`.
  - **allowDrawingsAsSpatialFilter**: When this property is `true`, users can limit the query to enclosed areas that are drawn on the map. For example, a user could draw several circles on the map and query only those features that lie within the circles. If you set this property to `false`, the option to constrain query operations to polygon drawings is not available to end users.  
You cannot use query tokens in this format string.
  - **queryProviderSupportsTimeOfDay**: If the data provider that you are using supports storing the time of day using Unix time, set this property to `true`. You cannot use query tokens in this format string.
  - **mode**: The `mode` for this view is `query`. You cannot use query tokens in this format string.
  - **numberOfSuggestions**: The number of suggestions displayed by the autocomplete function of the value expression field. The default value is `10`.
  - **defaultLogicalOperator**: The logical operator to be displayed when the query builder is initialized. The default value is `and`.
- **SimpleFilterBuilderView**: This view is used when the end user selects the Filter Builder.
    - **wildcard**: The character that your data provider uses as a wildcard. For example, the ANSI SQL default wildcard is `%`. You cannot use query tokens in this format string.
    - **dateQueryFormat**: The date format that your data provider uses. The syntax of the date format is `{0}: [date format]`, where `0` is the field name and `[date format]` is the date format that your data provider uses. Filter Builder uses the `dateQueryFormat` property to construct date queries.  
Your data provider's date format can contain any of the [.NET date and time format specifiers](#) that are listed below. For example, the date format for the ANSI SQL default data provider is `DATE: '{0:yyyy-MM-dd}'`. This uses three .NET format specifiers: `yyyy`, `MM`, and `dd`.
 

d	ffffff	h	s
dd	F	hh	ss
f	FF	H	y
ff	FFF	HH	yy
fff	FFFF	m	yyy
ffff	FFFFF	mm	yyyy
fffff	FFFFFF	M	yyyyy
ffffff	FFFFFFF	MM	
  - **textComparisonQueryFormat**: The format of a query that compares the value of a text field to the input value. For example, the format for text comparisons in the ANSI SQL default data provider is `LOWER`

`{0} LIKE LOWER({1})`.

- **numberToTextComparisonQueryFormat**: The format of a query that compares the value of a numeric field to the input value. For example, the format for number-to-text comparisons in the ANSI SQL default data provider is `CAST({0} AS VARCHAR(50)) LIKE '{1}'`.
- **doesNotContainQueryFormat**: The format of a query that tests whether a field contains the input value. For example, the format for text comparisons in the ANSI SQL default data provider is `LOWER({0}) NOT LIKE LOWER({1})`.
- **allowDrawingsAsSpatialFilter**: When this property is `true`, users can limit the query to enclosed areas that are drawn on the map. For example, a user could draw several circles on the map and query only those features that lie within the circles. If you set this property to `false`, the option to constrain query operations to polygon drawings is not available to end users.  
You cannot use query tokens in this format string.
- **queryProviderSupportsTimeOfDay**: If the data provider that you are using supports storing the time of day using Unix time, set this property to `true`. You cannot use query tokens in this format string.
- **mode**: The mode for this view is `filter`. You cannot use query tokens in this format string.
- **numberOfSuggestions**: The number of suggestions displayed by the autocomplete function of the value expression field. The default value is `10`.
- **defaultLogicalOperator**: The logical operator to be displayed when the filter builder is initialized. The default value is `and`.

- **ListQueriesView**: None
- **ListFiltersView**: None
- **SaveQueryView**: None
- **SaveFilterView**: None
- **QueryActionsView**:
  - **menuID**: The menu ID for the view. The default value is `"QueryActions"`.
- **FilterActionsView**:
  - **menuID**: The menu ID for the view. The default value is `"FilterActions"`.

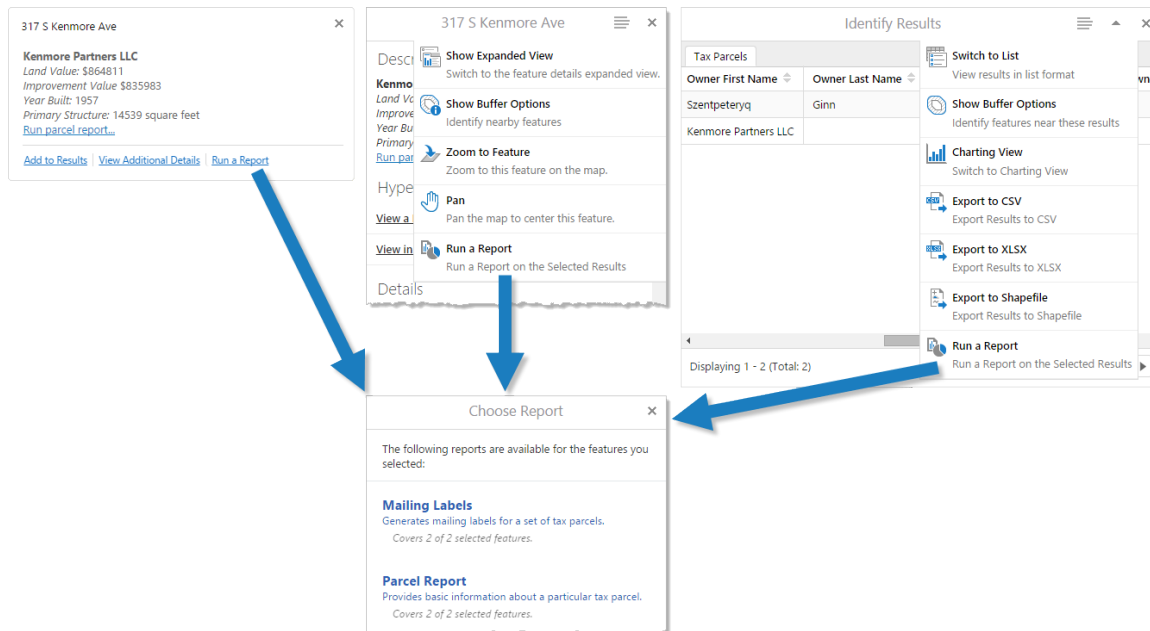
## View Models

- **SimpleQueryBuilderViewModel**: None
- **SimpleFilterBuilderViewModel**: None
- **ListQueriesViewModel**:
  - **mode**: The mode for this view model is `"query"`. You cannot use query tokens in this format string.
- **ListFiltersViewModel**:
  - **mode**: The mode for this view model is `"filter"`. You cannot use query tokens in this format string.

- **SaveQueryViewModel:**
  - **mode:** The mode for this view model is "query". You cannot use query tokens in this format string.
- **SaveFilterViewModel:**
  - **mode:** The mode for this view model is "filter". You cannot use query tokens in this format string.
- **QueryActionsViewModel:**
  - **mode:** The mode for this view model is "query". You cannot use query tokens in this format string.
- **FilterActionsViewModel:**
  - **mode:** The mode for this view model is "filter". You cannot use query tokens in this format string.

## 15.60 Reporting Module

The Reporting Module implements the ability to run reports in the HTML5 Viewer. If a layer has reports configured, the user can run reports for a feature (or features) within it from a link in a [map tip](#), [Results List](#) or [Results Table](#). For example, a user might use the Identify tool and run a report on the resulting features. For information about configuring reports, see "Reporting" in the *Geocortex Essentials Administrator Guide*.



The user can run a report from a map tip (left), Results List (middle) or Results Table (right)

## Configuration Properties

Module

None

## Views


- **ListReportsView:** None
- **RunReportView:** None

## View Models

- **ListReportsViewModel:** None
- **RunReportViewModel:** None

## 15.61 Results Module

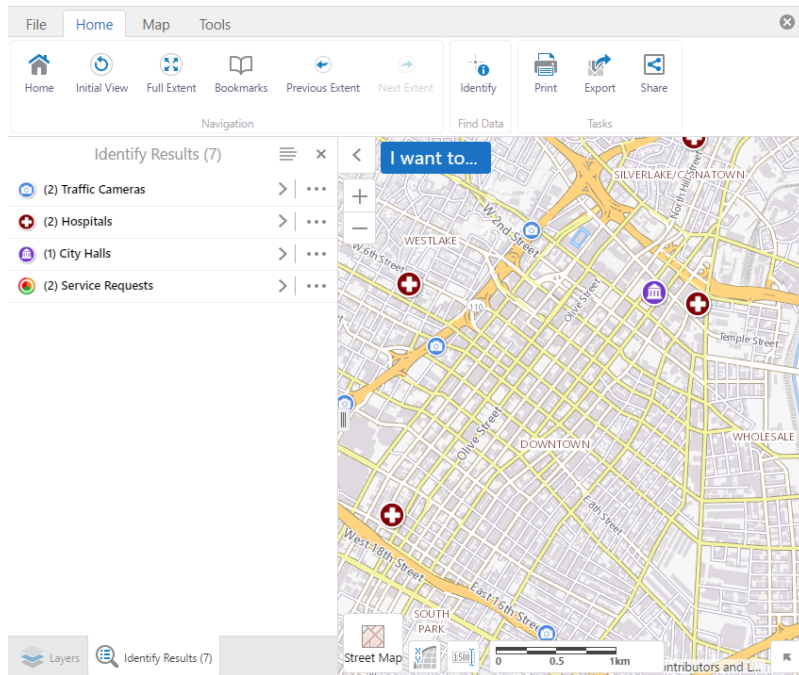
The Results Module provides views that display feature data. The Results Module's views honor the configuration defined in a site for layers and fields (visibility, aliases, and so on).

The Results Module can display results in two modes: a List view and a Table view. In the List view, results are displayed as a list in the side panel. In the Table view, results are displayed as a table in the bottom panel. The user can switch between the different views via the Panel Actions Menu  at the top-right of the panel. By default, results are displayed in the List view. The Handheld interface does not support the Table view. The List view can be activated via the `ShowResultsList` command, while the Table view can be activated via the `ShowResultsTable` command.



From the Management Pack, you can configure which results view the user should see by default. For information see [Change the Look and Feel on page 74](#).

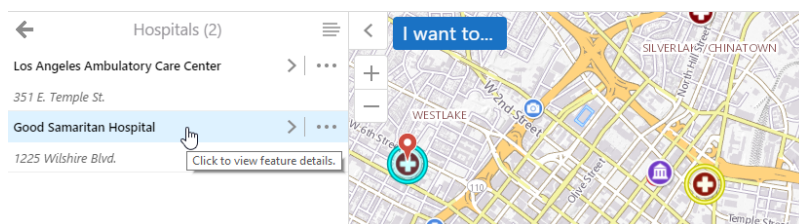
When a user completes an identify action, the view summarizes the results for each identified layer in a list in the side panel.



### Summarized list of identified results

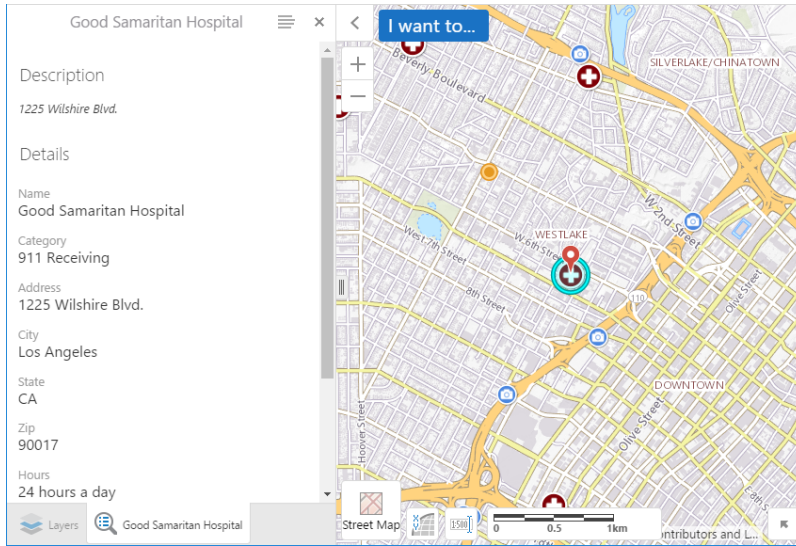
When the user clicks one of the summarized items in the list, the specific items are listed and the panel heading indicates the layer name and the number of items. Each item comprising the summary is listed.

For example, clicking **(2) Hospitals** in the summary list opens a list identifying the 2 hospitals. The hospitals are highlighted on the map, and when the user places the cursor on one of the entries, the corresponding hospital icon is graphically emphasized on the map.



### Item highlighted in the list and the icon emphasized on the map

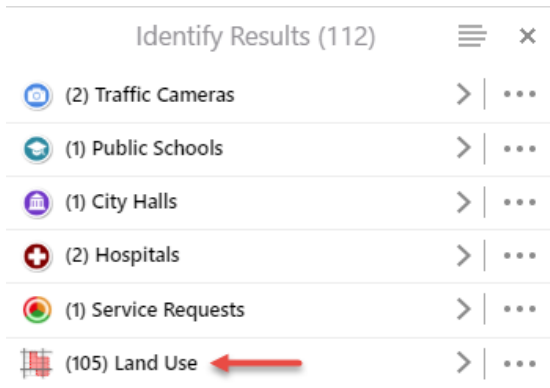
When the user clicks one of the hospital names, its details are displayed, and the corresponding icon for the hospital is emphasized and centered on the map.



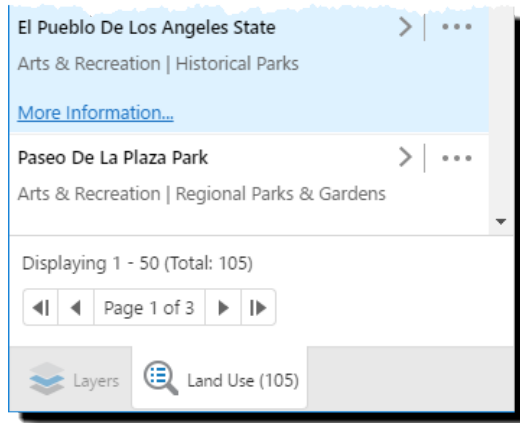
**Item details shown and the icon is emphasized and centered on the map**

When the user clicks an identified layer in the summary list that has more than 50 items, the specific items are listed in groups of 50 (the default), and a paging area opens at the bottom of the list panel. The total number of items is also shown. The user can page forward and backward, and go to the last page or return to the first page of the items list. As the user pages through the list, the paging area shows which group of 50 is being displayed, for example 1 - 50, 51 - 100, 101 - 150, and so on.

Note that the default of 50 items per page can be changed by updating `pageSize` in the `FeatureSetResultsView` (see Views in the Configuration Properties below).



**Land Use in the summary list indicating 105 items found**




### Specific Land Use items and the paging area below the list

When the user selects Switch to Table in the Panel Actions Menu, The view displays results in a table within separate tabs for each category, as illustrated below. Clicking a tab shows the data for that category. Using hospitals as an example, the **Hospitals** tab shows the details for each entry, and the hospitals are highlighted on the map. When the user moves the cursor over one of the entries, that hospital icon is graphically emphasized on the map. When the user clicks one of the hospital names, the corresponding icon for the hospital is centered on the map.

Name	Category	Address	City	State	Zip	Hours
Los Angeles Ambulatory Care Center		351 E. Temple St.	Los Angeles	CA	90012	Monday through Friday, 7:30a
Good Samaritan Hospital	911 Receiving	1225 Wilshire Blvd.	Los Angeles	CA	90017	24 hours a day

### Example of Results Table

The content of the Panel Actions Menu  is determined by the view that is open. The menu can contain actions for some or all of the following functions:

- Zooming to features
- Buffering
- Charting
- Exporting the results' attributes to a .csv file, an .xlsx file, or a shapefile (.shp file)

- Opening, saving, or combining results
- Running reports

The actions menus are configured in the [Menu Module](#)'s `ResultsListActions` menu and `ResultsTableActions` menu.

The Results Module does not need to be aware of which module is the source of the feature set collection that it is going to host, that is, which module generated the feature set collection and how it was requested. The request can originate from an identify, search, query, measurement, or map tip operation. You can configure the way the result of an operation is handled (that is, which commands are executed) by using the `resultMappings` element.

## Export Results with Data Links

If the results that a user wishes to export have data link data, an additional field (called `GCX_OID` by default) is added to the exported results data.

If the user exports results to CSV, the data link tables are included as an additional CSV file in the `Export.zip`. The `GCX_OID` correlates the data link's CSV and the exported results CSV, as it is a matching field in both files.

Similarly, if the user exports results to XLSX, the data link tables are included as additional sheets in the XLSX file included in the `Export.zip`. The `GCX_OID` correlates the data link's sheet and the exported results's sheet, as they have matching fields on each sheet.

## Export Results as Shapefiles

If the user exports results to a shapefile, an `Export.zip` is created with five shapefile pieces (`.shp`, `.shx`, `.dbf`, `.prj`, and `.cpg`).

When the results include data link data, additional shapefile pieces are created. Because data link data often does not include geographical data, only a subset of shapefile pieces are generated (`.dbf`, and `.cpg`).

## Configuration Properties

Module:

- **resultMappings:**
  - **Identify:** An array of commands to execute when the feature set collection was generated from an identify operation. The default is the set of commands: `ShowResultsList` and `SetCollectionOfInterest`.
  - **MapTip:** An array of commands to execute when the feature set collection was generated from a map tip operation. The default is the command `ShowMapTipResults`.
  - **Measurement:** An array of commands to execute when the feature set collection was generated from a measurement operation. By default, the array is empty. If you add the `ShowResultsTable` command to the array, measurements are presented in the Results Table, in addition to on the map. For information, see [Measurement Module on page 269](#).
  - **Coordinates:** An array of commands to execute when the feature set collection was generated from a plot coordinates operation. By default, the array is empty. If you add the `ShowResultsTable` command



---

to the array, coordinates are presented in the Results Table, in addition to on the map. For information, see [PlotCoordinates Module on page 297](#).

- **Workflow:** An array of commands to execute when the feature set collection was generated from a select features activity in a workflow. The default is the set of commands: `ShowResultsList` and `SetCollectionOfInterest`.
- **Search:** An array of commands to execute when the feature set collection was generated from a search operation. The default is the set of commands: `ShowResultsList` and `SetCollectionOfInterest`.
- **QueryBuilder:** An array of commands to execute when the feature set collection was generated from a QueryBuilder operation. The default is the set of commands: `ShowResultsList` and `SetCollectionOfInterest`.
- **ClusterFeatures:** An array of commands to execute when the feature set collection was generated from a cluster operation. The default is the set of two commands: `ShowMapTipResults` and `SetCollectionOfInterest`.
- **Selection:** An array of commands to execute when the feature set collection was generated from a Saved Results operation. The default is the set of commands: `ShowResultsList` and `SetCollectionOfInterest`.
- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **ResultsListFeatureClickedBehavior:** A behavior that runs an array of commands when a feature in the Results List is clicked. By default, this includes the command: `ShowFeatureDetails`.
  - **ResultsListFeaturePressedBehavior:** A behavior that runs an array of commands when a feature in the Results List is pressed for a long time. By default, this includes the command: `ShowFeatureDetails`.
  - **ResultsTableFeatureClickedBehavior:** A behavior that runs an array of commands when a feature in the Results Table is clicked. By default, this includes the following commands: `ShowMapTip`, `ZoomToFeature` and `HighlightFeatureDefault`.
  - **ResultsTableFeaturePressedBehavior:** A behavior that runs an array of commands when a feature in the Results Table is pressed for a long time. By default, this includes the command `ZoomToFeature`.
  - **FeatureSetClickedEventBehavior:** A behavior that runs an array of commands when a feature set in the Results View is clicked. By default, this includes the command `ShowFeatureSetResultsView`.
  - **ResultsListFeatureHoverBehavior:** A behavior that runs an array of commands when the cursor hovers over a feature in the Results List. By default, this includes the command `FocusFeature`.
  - **ResultsListFeatureHoverEndBehavior:** A behavior that runs an array of commands when the cursor no longer hovers over a feature in the Results List. By default, this includes the command `ClearFeatureFocus`.
  - **ResultsTableFeatureHoverBehavior:** A behavior that runs an array of commands when the cursor hovers over a feature in the Results Table. By default, this includes the command `FocusFeature`.
  - **ResultsTableFeatureHoverEndBehavior:** A behavior that runs an array of commands when the cursor no longer hovers over a feature in the Results Table. By default, this includes the command `ClearFeatureFocus`.

- **ResultsViewClosedBehavior:** A behavior that runs an array of commands when the Results View is closed – when the user clicks the X (Close Panel).
- **ResultsViewCollapsedBehavior:** A behavior that runs an array of commands then the Results View is collapsed – when the user clicks the < (Collapse Panel) icon beside the panel.
- **ResultsPageChangeBehavior:** A behaviour that runs an array of commands each time the user changes pages in the Results View. By default, this includes the command `ClearAndHighlightFeatures`.



You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.



You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.



Adding a new behavior is only recommended for experienced developers.

- **customSearchSuggestions:** Specifies the message to display to the user when an operation does not return any results. If you do not configure custom search suggestions, the viewer displays a built-in message.
  - **Identify:** Specifies the message to display when no results are returned for an identify operation. The built-in message for identify operations is **No results to display**.
  - **MapTip:** Specifies the message to display when no results are returned in a map tip. The built-in message for map tips is **No results**.
  - **Search:** Specifies the message to display when no results are returned for a search operation. The built-in message for search operations is **No results to display**.
  - **QueryBuilder:** Specifies the message to display when no results are returned for a QueryBuilder operation. The built-in message for QueryBuilder operations is **No results to display**.
  - **Workflow:** Specifies the message to display when no results are returned for a search, identify, map tip, or query operation that is run in a workflow. The built-in message for workflows is **No results to display**.
- **showInvisibleAttributesOnExport:** Specifies whether exported results tables should respect the visibility of fields. The default value is `false`.
- **includeDataLinksOnExport:** Specifies whether datalink data is exported in addition to results data. The default value is `true`.
- **enableDataLinkObjectIdFieldOnExport:** Specifies whether feature set collections with datalinks should include an Object ID field when results are exported. The default value is `true`.
- **gcxObjectIdFieldNameOnExport:** Specifies the format of the name of the Object ID field that is included when feature set collections have results with datalinks. The default field name is `GCX_OID`.

---

## Views

- **ResultsListView:**
  - `contentField`: Which field should be displayed as the content for a feature in the search Results List. Possible values include: `description` and `longDescription`. The default is `longDescription`.
  - `automaticDrillInDelay`: If the user performs an operation that returns results belonging to the same feature set in less time than this value, then individual results are listed in the Results list. If the operation that returns results takes longer than the drill-in delay setting, then the results are displayed in the summary panel, and the user needs to click the entry to open the list of individual results. The default drill-in delay setting is 300 milliseconds. If the setting is changed to 0, `automaticDrillInDelay` is disabled.
- **FeatureSetResultsView:**
  - `contentField`: Which field should be displayed as the content for a feature in the search Results List. Possible values include: `description` and `longDescription`. The default is `longDescription`.
  - `isPaged`: When set to `false`, all the items listed appear in a single scrollable page. When set to `true`, the list items are displayed in pages. The number of items per page depends on the page size, which is also configurable.
  - `pageSize`: The number of list items that should appear in a single page.
  - `showBackButton`: Shows the back button to go back to the ResultsListView.
  - `highlightMode`: Highlight mode is specified in the Management Pack, and the mode selections are: **Highlight Features On Page** (this is the default), **Highlight All Features**, and **Do Not Highlight Any Features**. Note that changing this field in the configuration file does not affect the highlight behaviour. See "Change the Look and Feel of an HTML5 Viewer" in the *Geocortex Essentials Administrator Guide*.
- **ResultsTableView:**
  - `isPaged`: When set to `false`, all the items listed appear in a single scrollable page. When set to `true`, the list items are displayed in pages. The number of items per page depends on the page size, which is also configurable.
  - `pageSize`: The number of list items that should appear in a single page.

## View Models

- **ResultsListViewModel**: None
- **FeatureSetResultsViewModel**: None
- **ResultsTableViewModel**: None

## Example: Add a Command with a Parameter to a Behavior

The following example demonstrates adding a new command with a parameter to the behavior, `ResultsListFeatureClickedBehavior`.

► **To add a command with a parameter to a behavior:**

1. Run an XML editor or text editor as an administrator.
2. Open one of the viewer configuration files, `Desktop.json.js`, `Tablet.json.js`, or `Handheld.json.js`, in the editor.

By default, the configuration files are here:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\[instance]\REST
Elements\Sites\[site]\Viewers\[viewer]\VirtualDirectory\Resources\Config\Default\
```

3. In the `Results` module section, find the `behaviors` property. Locate the behavior you want to edit. For example, `ResultsListFeatureClickedBehavior`.

```
{
  "moduleName": "Results",
  ...
  "configuration": {
    ...
    "behaviors": [
      {
        "name": "ResultsListFeatureClickedBehavior",
        "event": "ResultsListFeatureClickedEvent",
        "commands": [
          "ShowFeatureDetails"
        ]
      },
      ...
    ]
  },
  ...
}
```

The behavior executes a single command: `ShowFeatureDetails`.

4. Consult the Geocortex SDK for HTML5 API Reference to determine the type of parameter that is associated with this command or the behavior's event. Although `ShowFeatureDetails` accepts a parameter of either type, `geocortex.essentialsHtmlViewer.mapping.infrastructure.Feature` or `geocortex.essentialsHtmlViewer.mapping.infrastructure.FeatureSetCollection` ID (string),

`ResultsListFeatureClickedEvent` has a parameter of type, `geocortex.essentialsHtmlViewer.mapping.infrastructure.Feature`, so this is the type to use.

5. Find a command that you want to add to the behavior in the Geocortex SDK for HTML5 API Reference that has the same parameter type. For example, `PanToFeature`.

6. Add the desired command to the list of commands, separated by a comma. For example:

```
"commands": [
  "ShowFeatureDetails",
  "PanToFeature"
]
```

7. Save the file.

#### See Also...

[Menu Module on page 282](#)

[Measurement Module on page 269](#)

## 15.62 Scalebar Module



This module can be configured using Manager. For instructions, see [Configure Map Widgets on page 90](#).

The Scale Bar Module implements the scale bar control in the bottom left corner of the map.



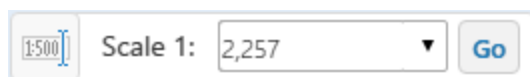
Dual-unit Line scale bar (left) and metric Ruler scale bar (right)

### Scale Input Box


The scale input box allows users to enter a new scale to zoom the map to. This feature always represents scale as a ratio. For example, 1:1128, 1:300890, or 1:11024035.



The scale input box is enabled by default and can be toggled by selecting the scale input icon in the map region's bottom left-hand corner. When toggled, the scale input icon expands to reveal the scale input box.

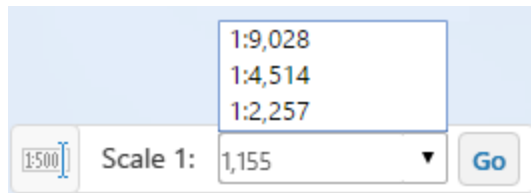


When submitting a custom scale input with the keyboard, the Viewer automatically adjusts the user input to the nearest available scale if the map uses a cached tiled basemap.

Use the up or down arrow  to adjust the scale by increments of one.



This feature is only available in Chrome for desktop.



Alternatively, if a map includes a cached tiled basemap, use the drop-down menu to pick from a preset list of scales.

To zoom to the new scale value, use the Enter key or press the **Go** button.

## Configuration Properties

### Module

None

### Views

- **ScaleInputBoxButtonView**: To hide the scale input box toggle button, set this view's `visible` property to `false`.
- **ScaleInputBoxView**: To show the scale input box, set this view's `visible` property to `true`.
- **ScalebarView**: To show the scale bar, set the view's `visible` property to `true`. The default value is `false`.
  - **scalebarStyle**: The style of the scale bar. The possible values are: `ruler` or `line`. The default is `ruler`.
  - **scalebarUnit**: The type of units of the scale bar. The possible values are: `metric`, `english` (US customary units), or `dual` (both metric and US customary units). The default is `metric`.



The `dual` option can only be used with the `line` scale bar style.

- **showBackground**: To display a background for the scale bar, set to `true`; otherwise, set to `false`. The default is `true`.

### View Models

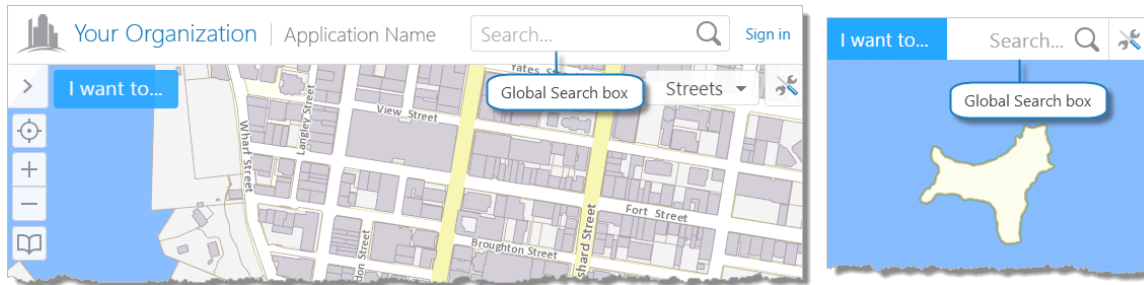
- **ScalebarViewModel**:
  - **scaleInputBox**: Configure the scale input box and its visibility.
    - **isEnabled**: Enables the scale input box. The default value is `false`.
    - **openByDefault**: Sets whether the scale input box is expanded by default. The default value is `true`.



If `isEnabled` is set to `false`, the `openByDefault` value is ignored.

## 15.63 Search Module

The Search Module implements Global Search, including the Global Search box and the search providers. By default, in the Desktop and Tablet interfaces, the Global Search box is located in the top right-hand corner of the viewer, in the banner region (BannerContentRegion). In the Handheld interface, the Global Search box is located at the top of the screen, in the HeaderRegion.




Global Search box in the HTML5 viewer's Desktop interface (left), and Handheld interface

The Global Search feature allows end users to search for features that match a search term entered by the user. For more information, see the "Global Search" section in the *Geocortex Essentials Administrator Guide*.


Global Search is capable of searching multiple sources. The results are aggregated and displayed in the Results List or Results Table, depending on how you configure the `resultMappings` property in the Results Module. For more information, see [Results Module on page 328](#).


You can configure the sources that are searched. There are two source types:

- **Geocoding Services:** You can configure the HTML5 viewer to search the geocoders that are configured in the site. Searching the geocoders is done in addition to searching any search providers that are configured.

 As of version 2.5, the HTML5 Viewer supports both single-line and multi-line ArcGIS geocoding services.

- **Search providers:** The HTML5 Viewer has two search providers: `LayerQuerySearchProvider`, which performs searches by sending search requests to the map services (Layer Search), and `InstantSearchProvider`, which searches the Instant Search index. The search providers search only those layers that have Global Search turned on. For instructions on configuring Global Search and Instant Search, refer to the "Global Search" section in the *Geocortex Essentials Administrator Guide*.

 You can navigate between search hints by pressing the **up arrow** (↑) or **down arrow** (↓) keys.

 WMS layers that are not associated with a WFS do not support search operations.

## Configuration Properties

### Module

- **autoLoadSiteGeocoders:** When this property is `true`, the viewer automatically loads the geocoders that are configured in the site. When the user does a global search, the geocoders are searched in addition to the search providers. The default is `true`. For information on configuring geocoders, refer to the "GIS Services" section of the *Geocortex Essentials Administrator Guide*.
- **searchProviders:** An array of search providers.
  - **LayerQuerySearchProvider:** None
  - **InstantSearchProvider:**
    - **maxResults:** The maximum number of search results to show.
    - **maxHints:** The maximum number of search hints to display.
    - **precedenceToNearbyResults:** When this property is `true`, search results are ranked by how close they are to the center of the current map and relevance to search terms. When the property is `false`, search results are ranked only by relevance to search terms. By default, this property is `true`.



This property replaces the `searchWithinCurrentExtentOnly` property found in earlier versions of the Geocortex Viewer for HTML5, which limited the search area within the current extent.

- **enableSearchHints:** When this property is `true`, the viewer suggests search terms to the user. The user can click a hint to search for the term given in that hint. By default, this property is `true`. When this property is `false`, the viewer does not display search hints.

### Views

- **SearchHintsView:** (Handheld only) None

### View Models

- **SearchViewModel:**
  - **enableSearchRefinement:** When this property is `true`, search hints have a Refine Search icon that the user can click to copy the search hint to the Global Search box. Because search hints are usually more complex than the search term, search hints refine the search. By default, this property is `true`. When this property is `false`, search hints do not have a Refine Search icon.
  - **delayConsecutiveSearches:** To delay consecutive searches for the number of milliseconds specified by `consecutiveSearchDelay`, set to `true`; otherwise, set to `false`. This is used to prevent too many concurrent requests. By default, this property is `false`.
  - **consecutiveSearchDelay:** The number of milliseconds to delay consecutive searches, when `delayConsecutiveSearches` is set to `true`. This is used to prevent too many concurrent requests. By



default, this property is omitted. If omitted, the default is 10000 milliseconds.

- **minimumPopulateDelay:** The number of milliseconds to wait after the user stops typing before search hints are displayed. The default is **300**.
- **minimumPrefixLength:** The number of characters the user must type before search hints are displayed. The default is **3**.

**See also...**

[Results Module on page 328](#)

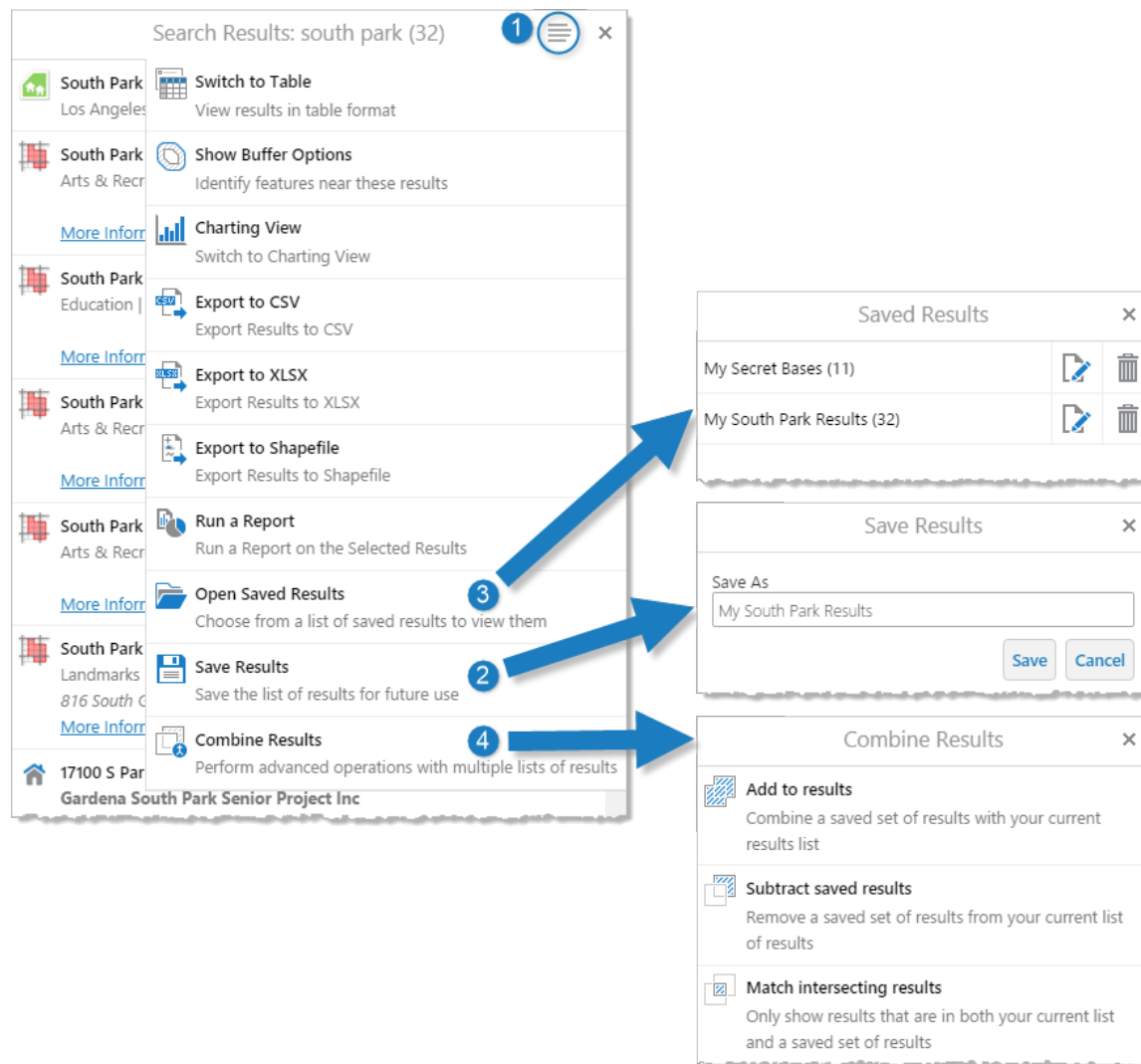
## 15.64 Selection Module

The Selection Module implements the Saved Results feature, which allows users to save, open and modify collections of features. After performing a search, query or identify operation, you can save the results with a name that appears in the title of the panel when the saved results are viewed. You can rename or delete saved results. You can only open one set of saved results at a time. You can add or remove features either through the Panel Actions Menu while viewing feature details, or via a link in a feature's map tip. You can also modify the current results by using the **Combine Results** option to either: add saved results, subtract saved results, or match intersecting saved results. Saved results that have unsaved changes are marked with an asterisk (\*) at the end of the panel title.



Saved Results only persist for the duration of your browser session. If you refresh or close the browser, you will lose all Saved Results.

To save results across sessions and share them with other users, a signed-in user can save a project, provided the viewer supports projects. For information about projects, see [Project Module on page 304](#).



In the Result List's Panel Actions Menu <sup>1</sup>, you can save the results <sup>2</sup>, and open them later <sup>3</sup>. You can also combine results <sup>4</sup>.

## Configuration Properties

Module:

- **resultsListStatusRegion:** The region to use for the Results List status message. This message informs the user when results have been saved.
- **resultsTableStatusRegion:** The region to use for the Results Table status message. This message informs the user when results have been saved.

- **menus:** An array of menus. By default, this includes a single menu: `CombineResultsActions`. Each menu has the following properties:
  - **id:** The menu's ID.
  - **description:** A short description of the menu.



If your viewer is going to be available in more than one language, enter the text key that the description is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **moduleId:** The ID of the module that the menu belongs to.
- **defaultIconUri:** The URI of the default icon for menu items.
- **items:** An array of menu items. Menu items have the following properties:
  - **iconUri:** The image to display beside the menu item. The recommended image size is 24px by 24px.
  - **text:** The text to appear on the menu item. You can use a text key or the literal text.
  - **description:** A brief description of the menu item to appear below the `text`. You can use a text key or the literal text.
  - **command:** The command to execute when the menu item is selected.



If you set the **command** property, do not set the **batch** property.

- **commandParameter:** The parameter value to pass to the command when it runs, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.



If you set the **commandParameter** property, do not set the **batch** property.

In the following example from the **CombineResultsActions** menu, a **commandParameter** is passed into the **CombineResultsInteractive** command to start the interactive process to combine results:

```
"command": "CombineResultsInteractive",
"commandParameter": {
  "mode": "union",
  "currentResults": "{{context}}"
}
```

The **{{context}}** token is a special token that retrieves the default menu context which a menu may have. For example, the default context of the `CombineResultsActions` menu is the `FeatureSetCollection` that represents the current results. Therefore, the command begins the interactive process of adding saved results to the current results.

- **hideOnDisable:** If this property is set to `true` and the menu item's command cannot run, the menu item does not show in the menu.  
If `hideOnDisable` is `false` and the menu item's command cannot run, the menu item shows in the menu, but it is grayed out.
- **batch:** An array of objects, each with three properties: an executable command, an optional command parameter, and an optional Boolean to abort the execution of further commands if a command fails to execute. This allows multiple commands to be executed in a specified order, although some commands may trigger asynchronous actions.  
The following example attempts to activate the Tabbed Toolbar and, if successful, proceeds to greet the user with an alert:

```
"batch": [  
  {  
    "command": "ActivateView",  
    "commandParameter": "TabbedToolbarView",  
    "abortBatchOnFailure": true  
  },  
  {  
    "command": "Alert",  
    "commandParameter": "Hello!"  
  }  
]
```

If omitted, `abortBatchOnFailure` is `false` by default.



If you set the `batch` property, do not set the `command` or `commandParameter` properties.

## Views

- **SaveSelectionView:** None.
- **ListSelectionsView:** None.
- **CombineResultsView:** None.

## View Models

- **SaveSelectionViewModel:** None.
- **ListSelectionsViewModel:**
  - `isPaged:` To paginate the Saved Results panel, set to `true`; otherwise, set to `false`. The default is `true`.
  - `pageSize:` The number of items per page. The default is 10.
- **CombineResultsViewModel:**

- `isPaged`: To paginate the Combine Results panel, set to `true`; otherwise, set to `false`. The default is `true`.
- `pageSize`: The number of items per page. The default is 10.

#### See Also...

[Project Module on page 304](#)

[Menu Module on page 282](#)

[FeatureDetails Module on page 168](#)

[MapTips Module on page 253](#)

## 15.65 Share Module

The Share Module implements the ability for end users to share a link to the viewer via email, social media, or any other type of web application. For example, if a user shares on Facebook, a post containing a link to the viewer is posted to the user's Facebook profile page. Anyone with access to the user's posts can click the link to open the viewer.

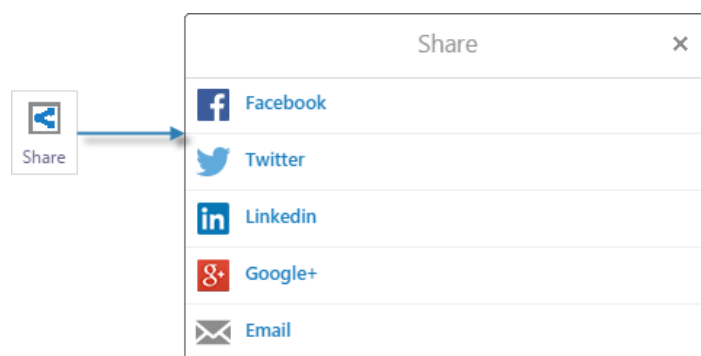
Sharing **options** are the application types (for example, email) or the specific applications (for example, Facebook) that the user can share links with. Options are configured using the Share Module's `shareOptions` property. By default, `shareOptions` are configured for Facebook, Twitter, LinkedIn, Google+, and email. You can configure additional options, provided the application's API supports sharing.

In order for end users to share the viewer's URL, you must provide a means to invoke the Sharing feature. For example, you could add sharing tools, hyperlinks, or I Want To menu items. The HTML5 Viewer has two different approaches to presenting sharing options:

- **List the Options:** Open a window that lists the sharing options for the user to choose from.
- **Provide Separate Options:** Provide a separate tool, menu item, or hyperlink for each sharing option.

### List the Options

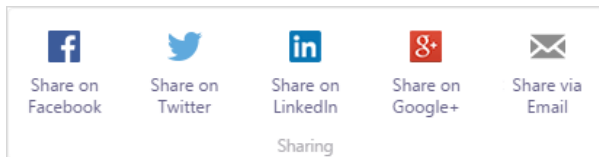
To list the sharing options, use the `ShowShareView` command, which displays the list of sharing options (the Share Module's `ShareView` view). The HTML5 Viewer has a Share tool that you can add to the toolbar to run the `ShowShareView` command.



The Share tool (left) lists the sharing options when clicked

## Provide Separate Options

The HTML5 Viewer has a `ShareOn` command that you can use to share with a specific `shareOption`.



### Example of a toolbar group with application-specific sharing tools


To add a sharing tool for a specific application, add a toolbar button that runs the `ShareOn` command with the `id` of the `shareOption` as the command parameter.

**Add Button**

Name:

Tooltip:   
@language-share-on-facebook

Hide on Disable:

Image Uri:  

Command:

Command Parameter:

### Example configuration to add a button to share on a specific platform

## Configuration Properties

### Module

- **shareOptions:** An array of applications to share the viewer's URL with. Each item has the following properties:
  - **id:** The ID of the application to share with. The ID must be unique across all of the `shareOptions`.
  - **displayName:** The name that displays in the `ShareView` view's list of sharing options.
  - **url:** The public URL that an application provides for sharing URLs. The `url` contains a placeholder, `{ViewerUrl}`, which is replaced at run time with the viewer's URL. The `SharingLink` Module is responsible for creating the URL that replaces `{ViewerUrl}`. For information, see [SharingLink Module on page 347](#).
  - **iconUri:** The relative URL of the image to show next to `displayName` in `ShareView`.

### Views

- **ShareView:** None

---

## View Models

- **ShareViewModel:**
  - **shareOptionsIds:** An array of the `shareOptions` to display in `ShareView`. If you omit `shareOptionsIds`, all the configured `shareOptions` are listed in `ShareView`.

### See also...

[SharingLink Module on page 347](#)

[Geocortex SDK for HTML5 API Reference on page 422](#)

## 15.66 SharingLink Module

The `SharingLink` Module creates viewer URLs that the `Share` Module can share with other applications.

As part of creating a viewer URL to share, the `SharingLink` Module assembles URL parameters. The URL parameters for sharing links fall into two categories:

- **Intrinsic:** URL parameters that are preserved from the URL that the user used to launch the viewer.
- **State-Preserving:** URL parameters that capture the viewer's current state.

### Intrinsic URL Parameters

Intrinsic URL parameters come from the URL that the user used to launch the viewer. Usually, intrinsic parameters are essential for the viewer to function properly. By default, the following URL parameters are intrinsic:

- `configBase`
- `viewer`
- `viewerConfigUri`

For example, suppose the original URL uses the `viewerConfigUri` parameter to point to the viewer's configuration files. When the `SharingLink` Module assembles a URL to share, it copies the `viewerConfigUri` parameter and its value from the original URL to the sharing URL that it is assembling.

You can override the default list of intrinsic parameters using the `intrinsicUrlParameters` property. Note that `intrinsicUrlParameters` **overrides** the default list—it does not add to it. This means that you must include the default intrinsic parameters in the list, as well as any other parameters that you want to preserve from the original URL.

For example, suppose the original URL contains the `viewerConfigUri`, `extent`, and `runWorkflow` parameters. Under the default configuration, the SharingLink Module copies `viewerConfigUri` to the sharing URL, but not `extent` or `runWorkflow`. If you want sharing links to run the workflow but not set the extent to the original URL's extent, you could configure `intrinsicUrlParameters` as follows:

```
{
  "moduleName": "SharingLink",
  "moduleType":
    "geocortex.essentialsHtmlViewer.mapping.modules.sharingLink.SharingLinkModule",
  "configuration": {
    intrinsicUrlParameters [
      configBase,
      viewer,
      viewerConfigUri,
      runWorkflow
    ],
    ...
  }
}
```

In this example, you do not have to include `viewer` and `viewerConfigUri` in `intrinsicUrlParameters`. However, it is a good practice to list all of the default intrinsic parameters.



We recommend that you list all the default intrinsic parameters in `intrinsicUrlParameters`, even if you do not use them all in the URLs that you distribute to your users. Then, if you ever change the method that you use to point to the viewer's configuration files, sharing links will still work.

The `intrinsicUrlParameters` property can include any of the parameters listed in [URL Parameters Reference on page 49](#).

## State-Preserving URL Parameters

State-preserving URL parameters preserve the state of the viewer when the user shares the link. Preserving the viewer's state ensures that other users see the same view of the map as the user who shared the link. For example, a URL parameter that captures the current map extent is state preserving.

The SharingLink Module creates state-preserving URL parameters when it assembles the sharing link. The specific parameters are determined by the `sharingLinkProviders` property. Each sharing link provider corresponds to a URL parameter that the SharingLink Module includes in the sharing link. The sharing link providers are:

- **LayerThemeSharingLinkProvider:** Preserves the **layer theme** that is active when the user shares the link.
- **LayersSharingLinkProvider:** Preserves a snapshot of which **layers** are visible when the user shares the link.
- **ExtentSharingLinkProvider:** Preserves the map **extent** that is in effect when the user shares the link.
- **CenterSharingLinkProvider:** Preserves the coordinates of the map's **center** point when the user shares the link.
- **ScaleSharingLinkProvider:** Preserves the map's **scale** when the user shares the link.

By default, the layer theme, layer visibility, center, and scale providers are enabled, and the extent provider is disabled. There is no need to enable the extent provider when the center and scale providers are enabled.



The SharingLink Module has a `HandleGenerateSharingLink` command that creates the URL. If a sharing link provider's `generate` property is `true`, `HandleGenerateSharingLink` creates the parameter for that provider and includes it in the URL.

Each sharing link provider also has an `apply` property, which applies the parameter when the site is first loaded. By default, `generate` and `apply` are `true`. To disable a sharing link provider, set `generate` and `apply` to `false`. If you want the end user to supply the URL parameter, set `generate` to `false` and `apply` to `true`.

Each sharing link provider has a `name` property that you can use to change the names of the state-preserving URL parameters that are included in sharing links. This is useful for localization. For example, if your users are predominantly French, you could set each provider's `name` property to the name's French translation. Users who click the shared link will see the French names for the URL parameters.

## Configuration Properties

### Module

- **`intrinsicUrlParameters`:** An array of URL parameters to include in the sharing link URL. By default, `intrinsicUrlParameters` includes `configBase`, `viewer`, and `viewerConfigUri`. The full list of possible parameters is given in [URL Parameters Reference on page 49](#).
- **`sharingLinkProviders`:** An array whose items control which aspects of the viewer's state are preserved in the sharing link. Each `sharingLinkProvider` corresponds to a URL parameter that can be included in the URL to share. Each provider has the following properties:
  - **`type`:** The type of sharing link provider.
  - **`name`:** The name that is used to identify the URL parameter within the URL. For example, in the URL `http://server.domain.com/vwr/Index.html?center=-13176043.9862,4002474.5385`, the parameter's name is **`center`**.
  - **`apply`:** When `apply` is `true`, the URL parameter is applied when the site is first loaded. If you omit `apply` from the configuration, it is `true`.
  - **`generate`:** When `generate` is `true`, the `HandleGenerateSharingLink` command creates the URL parameter for this provider and includes the parameter in the URL to share. If you omit `generate` from the configuration, it is `true`.

### Views

The SharingLinks Module does not have any views.

### View Models

The SharingLinks Module does not have any view models.

### See also...

[Share Module on page 345](#)

[Geocortex SDK for HTML5 API Reference on page 422](#)

[URL Parameters Reference on page 49](#)

## 15.67 Shells Module

The Shells Module implements the HTML5 Viewer's three interfaces ("shells") that enable the Viewer to work on different types of device. The three shells are:

- **Desktop:** For running the Viewer on desktop computers.
- **Tablet:** For running the Viewer on tablets.
- **Handheld:** For running the Viewer on handheld devices like smartphones.

The shells control the layout and the layout's behavior on the different types of device. A shell is a view that hosts a number of regions, along with some behavior to handle resizing and layout concerns.

### Configuration Properties

#### Module

- **css:** A list of file paths to the CSS file(s) for this shell. Each interface has its own CSS file to manage the differences in screen size and shape. The files are called `Desktop.css`, `Tablet.css`, and `Handheld.css`.
- **homePanelVisible:** When this property is set to `true`, the region that hosts the Home Panel, `HomePanelContainerRegion`, is visible when the viewer launches. If you want the user to be able to see the Home Panel when the viewer launches, set this property to `true`. The default is `false`.



To open the Home Panel when the viewer launches in the Desktop and Tablet interfaces, you must also set the `ShellViewModel` configuration property, `dataFrameOpenByDefault`, to `true`. This is not necessary for the Handheld interface.

#### Views

- **ShellView:**
  - **resizeShell:** Set to `true` for the Handheld interface. This parameter does not apply to the Desktop or Tablet interfaces.
- **DataFrameViewContainer:** None
- **ModalViewContainerView:** None
- **ResultsRegionViewContainerView:**
  - **resizableParentRegion:** The resizable parent region. The default is `BottomPanelRegion`.
  - **resizeY:** To allow the container to be vertically resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **FeatureEditingContainerView:**

- **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
  - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **LayerDataContainerView:**
    - **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
    - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **HomePanelContainerView:**
    - **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
    - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **DataFrameResultsContainerView:**
    - **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
    - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **ProjectContainerView:**
    - **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
    - **resizeX:** To allow the user to resize the container horizontally, set `resizeX` to `true`. If you do not want users to be able to resize the container horizontally, set `resizeX` to `false`. The default is `true`.
- **SimpleQueryBuilderContainerView:** None
    - **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
    - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **SimpleFilterBuilderContainerView:**
    - **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
    - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.
- **OfflineMapsContainerView:**
    - **resizableParentRegion:** The resizable parent region. The default is `LeftPanelRegion`.
    - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.

- **CoordinatesContainerView:**
  - `resizableParentRegion`: The resizable parent region. The default is `LeftPanelRegion`.
  - `resizeX`: To allow the container to be horizontally resized, set to true; otherwise, set to false. The default is true.
- **TimeSliderContainerView:**
  - `resizableParentRegion`: The resizable parent region. The default is `LeftPanelRegion`.
  - `resizeX`: To allow the container to be horizontally resized, set to true; otherwise, set to false. The default is true.
- **ProjectContainerView:**
  - `resizableParentRegion`: The resizable parent region. The default is `LeftPanelRegion`.
  - `resizeX`: To allow the container to be horizontally resized, set to true; otherwise, set to false. The default is true.
- **LayerAdditionContainerView:**
  - `resizableParentRegion`: The resizable parent region. The default is `LeftPanelRegion`.
  - `resizeX`: To allow the container to be horizontally resized, set to true; otherwise, set to false. The default is true.
- **DrawingOrderContainerView:**
  - `resizableParentRegion`: The resizable parent region. The default is `LeftPanelRegion`.
  - `resizeX`: To allow the container to be horizontally resized, set to true; otherwise, set to false. The default is true.
- **BottomPanelViewContainerView:** (Handheld interface only) None
- **ResultsViewContainerView:** (Handheld interface only) None
- **MiscContainerView:** (Handheld interface only) None
- **ModalContainerView:** (Handheld interface only) None

## View Models

- **ShellViewModel:**
  - `minWidth`: (Desktop and Tablet interfaces only) The minimum width to which the Data Frame can be resized. The default width is 200 pixels. Alternatively, you can specify the amount as a percentage string, for example, "10%".
  - `maxWidth`: (Desktop and Tablet interfaces only) The maximum width to which the Data Frame can be resized. The default width is 500 pixels. Alternatively, you can specify the amount as a percentage string, for example, "50%".
  - `dataFrameWidth`: (Desktop and Tablet interfaces only) The width of the Data Frame in pixels. The default width is 350 pixels. Alternatively, you can specify the amount as a percentage string, for example, "25%".



Specifying the amount as a percentage string ensures the panel size remains in proportion to the browser window until the user resizes the panel.

- **dataFrameOpenByDefault:** Controls whether the Data Frame is open when the viewer launches. The default is `false`.
- **bottomRegionHeight:** (Desktop and Tablet interfaces only) The height of the `ResultsRegion` in pixels. The `ResultsRegion` hosts the `FeatureDetailsView` and `ResultsTableView` by default. The default is 300 pixels. Alternatively, you can specify the amount as a percentage string, for example, "25%".



Specifying the amount as a percentage string ensures the panel size remains in proportion to the browser window until the user resizes the panel.

- **openToMaximum:** (Handheld interface only) To have the bottom panel open to the maximum size by default, set to `true`; otherwise, set to `false`. The default is `false`.
- **bottomPanelHeightPercent:** (Handheld interface only) The height of the bottom panel as a percentage. The default is 75.
- **DataFrameViewContainerViewModel:**
  - **defaultViewIcon:** The path to the icon to use if the view's `iconUri` is not specified.
  - **containerRegionName:** The name of the region in which this view container is hosted. The default is `DataRegion`.
  - **headerIsVisible:** Specifies whether or not the header in the view is visible. The default is `false`.
  - **showHeaderForStandaloneViews:** Specifies whether or not to display the header if the view is a stand alone view. The default is `true`.
  - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of 0 configured inside the node `ordering`). The default is `false`.
  - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
  - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `false`.
  - **resizeX:** To allow the container to be horizontally resized, set to `true`; otherwise, set to `false`. The default is `true`.
  - **footerInsertMarkup:** Is the URI of the markup file to be hosted in the footer of a view container.
  - **footerInsertType:** The view type to be hosted in the footer of a view container.
  - **ordering:** A mapping of views and their order ranking starting with 0. A view with the rank of 0 is the root view of a particular view container.

- **DataFrameResultsContainerViewModel:**

- **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `DataFrameResultsContainerRegion`.
- **statusRegionName:** The region to use for status messages in the Results List. The default is `DataFrameResultsStatusRegion`.
- **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
- **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `false`.
- **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `false`.
- **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
- **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.

- **LayerDataContainerViewModel:**

- **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `LayerDataContainerRegion`.
- **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
- **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `false`.
- **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `false`.
- **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
- **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.

- **HomePanelContainerViewModel:**

- **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `HomePanelContainerRegion`.
- **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
- **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `false`.
- **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `false`.
- **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
- **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.

- 
- **FeatureEditingContainerViewModel:**
    - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `FeatureEditingContainerRegion`.
    - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
    - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
    - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `false`.
    - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
  
  - **ModalViewContainerViewModel:**
    - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `ModalWindowRegion`.
    - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `false`.
    - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
    - **CloseOnEscape:** Whether or not the view closes when the user presses the **Esc** key. The default is `true`.
  
  - **ResultsRegionViewContainerViewModel:**
    - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `ResultsRegion`.
    - **statusRegionName:** The region to use for status messages in the Results Table. The default is `ResultsStatusRegion`.
    - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`).
    - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
    - **showMaximizeButton:** When set to `true`, the maximize button is displayed. When set to `false`, the maximize button is not displayed. The default is `true`.
    - **resizeY:** To allow the container to be vertically resized, set to `true`; otherwise, set to `false`. The default is `true`.
    - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
-

- **SimpleQueryBuilderViewContainerViewModel:**
  - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `SimpleQueryBuilderContainerRegion`.
  - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
  - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
  - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
  - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
  - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
  
- **SimpleFilterBuilderViewContainerViewModel:**
  - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `SimpleFilterBuilderContainerRegion`.
  - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
  - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
  - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
  - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
  - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
  
- **OfflineMapsContainerViewModel:**
  - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `OfflineMapsContainerRegion`.
  - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
  - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
  - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `false`.
  - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
  - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.



- 
- **CoordinatesContainerViewModel:**
    - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `CoordinatesContainerRegion`.
    - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
    - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
    - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
    - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
    - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
  
  - **TimeSliderContainerViewModel:**
    - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `TimeSliderContainerRegion`.
    - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
    - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
    - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
    - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
    - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
  
  - **ProjectContainerViewModel:**
    - **containerRegionName:** The name of the region in which this view container is hosted. The default is `ModalWindowRegion`.
    - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
    - **backButtonOnRootView:** When `backButtonOnRootView` is set to `true`, the back button displays on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `false`.
    - **showBackButtonAsX:** When `showBackButtonAsX` is set to `true`, the back button has X on it. When `showBackButtonAsX` is set to `false`, the button is displayed with `Back` written on it. The default is `true`.
    - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
    - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
-

- **LayerAdditionContainerViewModel:**
  - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `LayerAdditionContainerRegion`.
  - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
  - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
  - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
  - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
  - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.
- **DrawingOrderContainerViewModel:**
  - **containerRegionName:** Defines the name of the region in which this view container is hosted. The default is `DrawingOrderContainerRegion`.
  - **containerManagedTitle:** Defines the title of the bottom tab used when the Drawing Order panel is active in the viewer. The default value is a language key, `@language-layerDrawingOrder-tabTitle`.
  - **headerIsVisible:** Defines whether or not the header in the view is visible. The default is `true`.
  - **backButtonOnRootView:** When set to `true`, this parameter displays the back button on the root view (the view with the value of `0` configured inside the node `ordering`). The default is `true`.
  - **showBackButtonAsX:** When set to `true`, the back button has X on it. When set to `false`, the button is displayed with `Back` written on it. The default is `true`.
  - **showHostedViews:** Whether or not to display the views hosted in this view model. The default is `true`.
  - **ordering:** A mapping of views and their order ranking starting with `0`. A view with the rank of `0` is the root view of a particular view container.

## 15.68 SimpleLayerList Module



As of Geocortex Viewer for HTML5 2.3, the SimpleLayerList Module no longer exists. All of its functionality has been moved into the [LayerList Module](#).

### See Also...

[LayerList Module on page 232](#)

## 15.69 Site Module

The Site Module initializes an Essentials Site object and manages its life cycle.

### Configuration Properties

#### Module

- **siteUri:** The URI of the site.

#### Views

- **ServiceLayersFailureView:** None
- **SignInErrorView:** None

#### View Models

- **ServiceLayersFailureViewModel:** None

## 15.70 SkipLinks Module



This module can be configured using Manager. For instructions, see [Configure Accessibility on page 72](#).

The SkipLinks module allows keyboard users to open a skip navigation menu containing links to the most used areas in the Viewer. Users can traverse the links by tabbing.

This module is supported in the Desktop shell.

The default skip navigation link items are defined in `SkipLinksActions` and consists of links to the **Side Panel**, **Search, I want to...**, **Toolbar**, and **Map**. Initial focus is on the **Side Panel** link, and subsequent tabbing traverses each of the items in the menu. Users can quickly return to the skip navigation links menu by clicking the address field in the Viewer and then tabbing to open the menu. Users can also return to the initial item in the menu by continuing to tab through the Viewer, or they can access the last link in the menu by tabbing back (SHIFT+TAB) through the Viewer.

Note that only 5 skip link items are supported in order to minimize the amount of tabbing required to access specific areas in the Viewer.

## Configuration Properties

### Module

- **menus:** An array of items to display in the skip navigation links menu.
  - **id:** The ID for the item in the menu.
  - **defaultIconUri:** The URI of the default icon for menu items. This property has no effect on the skip navigation links menu in the Viewer. The default icon for the menu items is visible on the Accessibility page in the Management Pack if **iconURI** is not configured.
  - **items:** An array of menu items with the following properties
    - **iconUri:** The image associated with the menu item. This property has no effect in the skip navigation links menu in the Viewer. The icons are included on the menu items that are visible on the Accessibility page in the Management Pack.
    - **text:** The text for the menu item. You can use a text key or the literal text.
    - **description:** A brief description of the menu item.
    - **command:** The command to execute when the menu item is selected.
    - **commandParameter:** The parameter value to pass to the command when it runs, if it has a parameter.
    - **hideOnDisable:** If this property is set to true and the menu item's command cannot run, the menu item does not show in the menu.

### Views

- **SkipLinksView:** None

### View Models

- **SkipLinksViewModel:** None

### See Also...

[Menu Module on page 282](#)

[Accessibility on page 402](#)

## 15.71 Snapping Module



Internet Explorer 10, 11, and Microsoft Edge do not support snapping when editing point markup or features.

The Snapping Module provides snapping capabilities to various tools that work with geometries. When enabled, snapping allows the user to precisely draw shapes on the map by snapping to the nearest point, vertex or edge of an existing shape within a configurable radius around the mouse pointer. The user may enable or disable snapping by either clicking the **Enable Snapping** toggle button, or pressing the keyboard shortcut, which is **F** by default. The user may choose which layers are affected by snapping by clicking the **Select Snapping Layers** button.



For the keyboard shortcut to enable or disable snapping to work, the mouse pointer must be over the map.



The snapping feature snaps to the nearest point, vertex or edge to the mouse pointer in that order. For example, if snapping is enabled and a point feature, a vertex, and an edge are similarly near the mouse pointer, the application will snap to the point feature. Similarly, if there is only a vertex and an edge near the mouse pointer, the application will snap to the vertex.

The following tools support snapping: **Measure**, **Draw**, **Edit**, **Create New Feature**, and non-dragging **Identify** tools. The snapping controls are configured within both the [TabbedToolbar](#) and [CompactToolbar](#) modules in the context-sensitive toolbars associated with the following states: `MeasureState`, `DrawMarkupState`, `EditingMarkupState`, `FeaturePlacementGraphicState`, and `FindDataState`.



Snapping is only available for the Desktop interface; the Tablet and Handheld interfaces do not support snapping.



To configure which layers may be snapped to and which layers snapping applies to by default, edit the layer in Essentials Manager and configure the **Allow Snapping** and **Snapping Enabled** settings. For more information, see "Layer Settings" in the *Geocortex Essentials Administrator Guide*.



The easiest way to configure snapping for the **Measure**, **Draw**, **Edit**, **Create New Feature**, and non-dragging **Identify** tools is to edit your viewer in Essentials Manager, navigate to the **Toolbar** section, and ensure these tools are added along with **MeasurementToolControlRegion**, **MarkupEditRegion**, **EditControlRegion**, **CreateNewFeatureControlRegion**, or **FindDataControlRegion**, respectively.

## Configuration Properties

### Module

- **threshold**: The pixel radius around the mouse pointer where snapping occurs when enabled. The default is 25.
- **radiusFillColor**: The color of the area around the mouse pointer where snapping occurs when enabled. You can specify either a web color name or hexadecimal value. The default is #ffffff.
- **radiusFillOpacity**: The opacity of the area around the mouse pointer where snapping occurs when enabled. The maximum is 1 (completely opaque), and the minimum is 0 (completely transparent). The default is 0.2.
- **radiusBorderColor**: The border color of the area around the mouse pointer where snapping occurs when enabled. You can specify either a web color name or hexadecimal value. The default is #000000 (black).
- **radiusBorderSize**: The border size, in pixels, of the area around the mouse pointer where snapping occurs when enabled. The default is 1 pixel.
- **pointColor**: The color of the point where the next vertex is to be placed. You can specify either a web color name or hexadecimal value. The default is #ffffff (white).

- **pointSize:** The size of the point, in pixels, where the next vertex is to be placed. The default is 5 pixels.
- **toggleKey:** The JavaScript key code for the keyboard shortcut to enable or disable snapping. The default is 70 (which is the **F** key).



For a complete list of JavaScript key codes, [visit this website](#).

- **supportedDrawModes:** An array of the draw modes to support snapping. By default, they are POINT, MULTI\_POINT, LINE, POLYGON, POLYLINE.
- **snappingProvider:** A snapping provider with the following properties:
  - **graphicsLayers:** An array of graphic layers used by the snapping feature. By default, they are: Drawings\_measurement and Drawings.
- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **SelectLayersForSnappingActivatedBehavior:** A behavior that runs an array of commands when the user clicks the **Select Snapping Layers** button. By default, this includes the command: OpenDataFrame.
  - **SelectLayersForSnappingDeactivatedBehavior:** A behavior that runs an array of commands when **Snappable Layers** panel is dismissed by a toggle button. By default, this includes the command: CloseDataFrame.



You can remove or rearrange the commands of any behavior. You can also remove behaviors altogether.



You can add commands to a behavior if the command does not require a parameter, or if the type of the command's parameter matches the parameter of an existing command or the event associated with the behavior. To determine if the parameters are compatible, see the [Geocortex SDK for HTML5 API Reference](#). Note that private commands and events are not documented.



Adding a new behavior is only recommended for experienced developers.

## Views

- **SnappingLayerSelectorView:** None

## View Models

- **SnappingLayerSelectorViewModel:** None

## See Also...

[TabbedToolbar Module on page 363](#)

[CompactToolbar Module](#) on page 150

[Measurement Module](#) on page 269

[Markup Module](#) on page 258

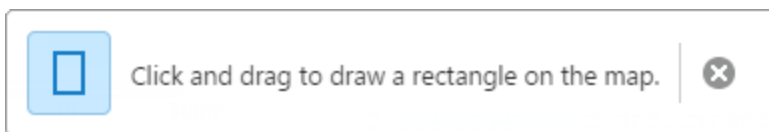
[Editing Module](#) on page 160

[Identify Module](#) on page 193

## 15.72 Status Module

The Status Module is a general-purpose module that can be used by any module to show a status message. There are generally two status types:

- Those that ask the user to perform an action. This message is shown with a static (non-animated) icon and text to instruct the user to perform a task, for example, to drag a rectangle on the map.



- Those that indicate that the application is currently working. This message is shown with an animated icon and a message that indicates that the application is working. For example, to show the map loading status.



### Configuration Properties

#### Module

- **busyIcon:** The URI of an image. It can be used to show a busy icon in the status message.

#### Views

- **StatusIndicatorView:** None
- **StatusMessageView:** None

#### View Models


- **StatusMessageViewModel:** None

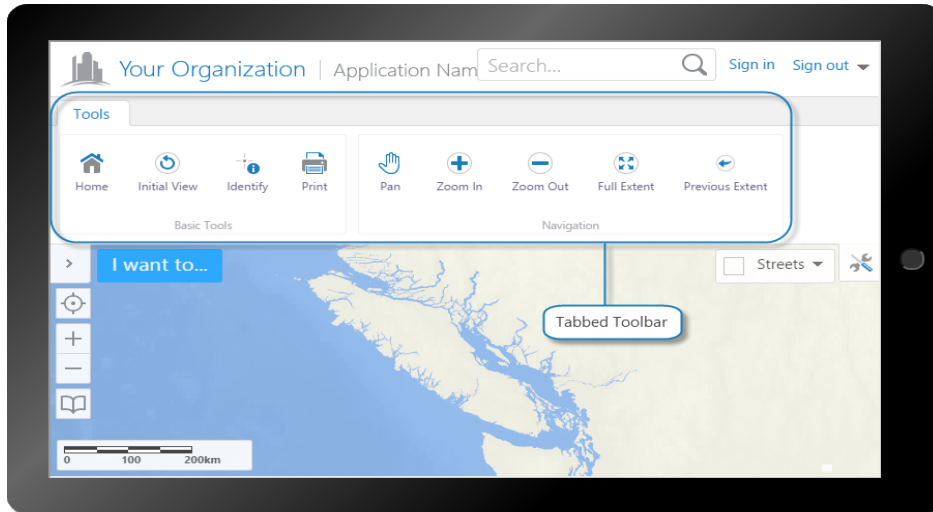
## 15.73 TabbedToolbar Module



The toolbar can be configured using Manager. See [Configure the Toolbar](#) on page 108 for instructions.

The TabbedToolbar Module implements the Tabbed Toolbar typically found in the Desktop and Tablet interfaces. When activated, the Tabbed Toolbar appears below the banner.

 As of version 2.4, the Handheld interface uses the Compact Toolbar by default. The Compact Toolbar is implemented by the [CompactToolbar Module](#). It is still possible to use the Tabbed Toolbar in the Handheld interface.

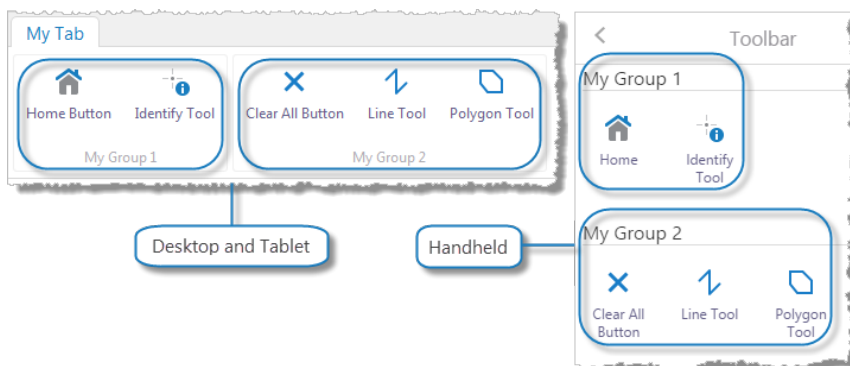


The Tabbed Toolbar, shown in the Tablet interface

## Tabbed Toolbar

The Tabbed Toolbar is made up of tabs, groups, buttons, tools, multitools (also known as flyouts), and a Tool Labels toggle button. Buttons immediately run commands that do not require input from the user. Tools run commands that operate on a geometry that the user draws; when the user clicks a tool, the tool must wait for the user to draw the geometry before running the command. Multitools act as menus of various related tools or buttons. The Tool Labels toggle button is provided by default and is not configurable. For more information about the Tool Labels toggle button, see [Configure the Toolbar on page 108](#).

Tools and buttons are arranged in groups, and groups are arranged on tabs. The following example includes a tab named **My Tab**, containing two groups: **My Group 1** and **My Group 2**. In the Handheld interface, tabs are ignored and groups are stacked on top of each other.






## A Tabbed Toolbar consisting of a single tab with two groups

In the configuration, tabs are configured as groups of groups. Whenever you configure a `toolbarGroup` that has at least one `toolbarGroup` item within it, the outer `toolbarGroup` is rendered in the viewer as a tab. The inner `toolbarGroup` items are rendered as groups within the tab. In the above example, the tab named **My Tab** is configured as a `toolbarGroup` that contains two other `toolbarGroup` items.

In the Desktop and Tablet interfaces, the Tabbed Toolbar has three views, all of which use the `TabbedToolbarViewModel`:

- **Tabbed Toolbar:** The `TabbedToolbarView` is used for the Tabbed Toolbar that appears immediately below the banner in the Desktop and Tablet interfaces, or in the `MiscViewContainerRegion` for the Handheld interface.
- **Toolbar Button:** The `TabbedToolbarButtonView` is used for the toolbar button  that the user clicks to open the Tabbed Toolbar. The toolbar button displays in the `ToolbarRegion` in the Desktop and Tablet interfaces, or in the `HeaderRegion` in the Handheld interface.
- **Toolbar Flyout:** The `ToolbarFlyoutView` is used to display the content of multitools. A multitool is a tool that contains multiple related tools.

## Configuration Properties

### Module

- **isEnabled:** To enable the Tabbed Toolbar, set to `true`; otherwise, set to `false`. The default is `false`.
- **transientElements:** An array of elements that define context-sensitive toolbars, each of which are associated with a [state](#), widget, region and a set of toolbar items.



As of HTML5 Viewer 2.5, each element must be associated with an application [state](#) to create context-sensitive toolbars.

- **stateName:** The name of the application state that triggers the context-sensitive toolbar.



For a complete list of states, see the [State Reference on page 452](#).

- **widgetId:** The ID of the widget.
- **region:** The name of the region to use.



For the context-sensitive menu to be available in the Tabbed Toolbar, the `region` must be added to the `items` array of a group in the `toolbarGroups` array.

- **items:** An array of toolbar items, each of which is either a button or toggle button.

## Properties of Buttons

- **id:** A unique ID for this button.
- **type:** The type is `button`.
- **iconUri:** The URI for the icon that you want to appear on the button. The image must be an appropriate size to fit on the button. Valid file formats are PNG, BMP, JPG, and JPEG.
- **command:** The command that the button runs when the user clicks the button.  
For information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters. For information on a particular command's parameter, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **hideOnDisable:** If this property is set to `true` and the button's command cannot run under the current configuration or run-time conditions, the button does not show in the toolbar. If `hideOnDisable` is `false` and the button's command cannot run, the button shows in the toolbar, but it is grayed out.
- **name:** The name that you want to appear on the button. You can use a text key or the literal text. For example, `@language-toolbar-home-sub` or `Home`.
- **tooltip:** The text for the tool tip that opens when the user positions the pointer over the button. You can use a text key or the literal text. For example, `@language-toolbar-navigation-home-tooltip` or `Returns to introductory page`.

## Properties of Toggle Buttons

- **id:** A unique ID for this `toggleButton`.
- **type:** The type is `toggleButton`.
- **toggleStateName:** (Optional) The name of the toggle [state](#) that this toggle button affects.
- **toggleOn:** Configures the toggle-on button, which turns the toggle button on:
  - **name:** The name that you want to appear on the toggle-on button. You can use a text key or the literal text. For example, `@language-toolbar-home-sub` or `Home`.
  - **tooltip:** The text for the tool tip that opens when the user positions the pointer over the toggle-on button. You can use a text key or the literal text. For example, `@language-toolbar-navigation-home-tooltip` or `Returns to introductory page`.
  - **iconUri:** The URI for the icon that you want to appear on the toggle-on button. The image must be an appropriate size to fit on the toggle-on button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable:** If this property is set to `true` and the toggle-on button's command cannot run under the current configuration or run-time conditions, the toggle-on button

does not show in the toolbar.

If `hideOnDisable` is `false` and the toggle-on button's command cannot run, the toggle-on button shows in the toolbar, but it is grayed out.

- **command**: The command that the toggle-on button runs when the user clicks the toggle button to turn it on.

For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).

- **commandParameter**: The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.

For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).

- **toggleOff**: Configures the toggle-off button, which turns the toggle button off:
  - **name**: The name that you want to appear on the toggle-off button. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-clear` or **Clear Markup**.
  - **tooltip**: The text for the tool tip that opens when the user positions the pointer over the toggle-off button. You can use a text key or the literal text.  
For example, `@language-toolbar-markup-clear-tooltip` or **Clear all drawings from the map**.
  - **iconUri**: The URI for the icon that you want to appear on the toggle-off button. The image must be an appropriate size to fit on the toggle-off button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable**: If this property is set to `true` and the toggle-off button's command cannot run under the current configuration or run-time conditions, the toggle-off button does not show in the toolbar.  
If `hideOnDisable` is `false` and the toggle-off button's command cannot run, the toggle-off button shows in the toolbar, but it is grayed out.
  - **command**: The command that the toggle-off button runs when the user clicks the toggle button to turn it off.  
For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).
  - **commandParameter**: The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).

- **toolbarGroups:** An array of `toolbarGroup` items, each of which is rendered as a tab in the Tabbed Toolbar.

### Properties of Tabs

- **id:** A unique ID for this `toolbarGroup`.
- **type:** The type is `toolbarGroup`.
- **name:** The name that you want to appear on the tab.  
For example, **@language-toolbar-group-tools** or **Tools**.



If your viewer is to be available in more than one language, enter the text key that the tab name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **items:** An array of `toolbarGroup` items, each of which is rendered as a group in the toolbar.

### Properties of Groups

- **id:** A unique ID for this `toolbarGroup`.
- **type:** The type is `toolbarGroup`.
- **name:** The name that you want to appear on the group. You can use a text key or the literal text.  
For example, **@language-toolbar-group-home** or **Basic Tools**.
- **items:** An array of toolbar items, each of which is either a button, toggle button, tool, region, or flyout (also known as a multitool).

### Properties of Buttons

- **id:** A unique ID for this `button`.
- **type:** The type is `button`.
- **iconUri:** The URI for the icon that you want to appear on the button. The image must be an appropriate size to fit on the button. Valid file formats are PNG, BMP, JPG, and JPEG.
- **command:** The command that the button runs when the user clicks the button.  
For information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **hideOnDisable:** If this property is set to `true` and the button's command cannot run under the current configuration or run-time conditions, the button does not show in the toolbar.  
If `hideOnDisable` is `false` and the button's command cannot run, the button shows in the toolbar, but it is grayed out.

- **name:** The name that you want to appear on the button. You can use a text key or the literal text.  
For example, **@language-toolbar-home-sub** or **Home**.
- **tooltip:** The text for the tool tip that opens when the user positions the pointer over the button. You can use a text key or the literal text.  
For example, **@language-toolbar-navigation-home-tooltip** or **Returns to introductory page**.

## Properties of Toggle Buttons

- **id:** A unique ID for this `toggleButton`.
- **type:** The type is `toggleButton`.
- **toggleStateName:** (Optional) The name of the toggle [state](#) that this toggle button affects.
- **toggleOn:** Configures the toggle-on button, which turns the toggle button on:
  - **name:** The name that you want to appear on the toggle-on button. You can use a text key or the literal text.  
For example, **@language-toolbar-home-sub** or **Home**.
  - **tooltip:** The text for the tool tip that opens when the user positions the pointer over the toggle-on button. You can use a text key or the literal text.  
For example, **@language-toolbar-navigation-home-tooltip** or **Returns to introductory page**.
  - **iconUri:** The URI for the icon that you want to appear on the toggle-on button. The image must be an appropriate size to fit on the toggle-on button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable:** If this property is set to `true` and the toggle-on button's command cannot run under the current configuration or run-time conditions, the toggle-on button does not show in the toolbar.  
If `hideOnDisable` is `false` and the toggle-on button's command cannot run, the toggle-on button shows in the toolbar, but it is grayed out.
  - **command:** The command that the toggle-on button runs when the user clicks the toggle button to turn it on.  
For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).
  - **commandParameter:** The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).

- **toggleOff**: Configures the toggle-off button, which turns the toggle button off:
  - **name**: The name that you want to appear on the toggle-off button. You can use a text key or the literal text.  
For example, **@language-toolbar-markup-clear** or **Clear Markup**.
  - **tooltip**: The text for the tool tip that opens when the user positions the pointer over the toggle-off button. You can use a text key or the literal text.  
For example, **@language-toolbar-markup-clear-tooltip** or **Clear all drawings from the map**.
  - **iconUri**: The URI for the icon that you want to appear on the toggle-off button. The image must be an appropriate size to fit on the toggle-off button. Valid file formats are PNG, BMP, JPG, and JPEG.
  - **hideOnDisable**: If this property is set to `true` and the toggle-off button's command cannot run under the current configuration or run-time conditions, the toggle-off button does not show in the toolbar.  
If `hideOnDisable` is `false` and the toggle-off button's command cannot run, the toggle-off button shows in the toolbar, but it is grayed out.
  - **command**: The command that the toggle-off button runs when the user clicks the toggle button to turn it off.  
For information on commands, see the [Geocortex SDK for HTML5 API Reference](#).
  - **commandParameter**: The value for the command to use as its parameter, if it has a parameter. Parameters may either be simple strings or complex objects containing any number of parameters.  
For information on a particular command's parameter, see the [Geocortex SDK for HTML5 API Reference](#).

## Properties of Tools

- **id**: A unique ID for this `tool`.
- **type**: The type is `tool`.
- **iconUri**: The URI for the icon that you want to appear on the tool. The image must be an appropriate size to fit on the tool. Valid file formats are PNG, BMP, JPG, and JPEG.
- **command**: The command that the tool runs after the user has drawn the geometry for the command to operate on.  
For information on commands, see [Geocortex SDK for HTML5 API Reference on page 422](#).
- **drawMode**: The type of geometry the user draws, upon which the tool operates.
- **name**: The name that you want to appear on the tool. You can use a text key or the literal text.  
For example, **@language-toolbar-tasks-identify** or **Identify**.
- **tooltip**: The text for the tool tip that opens when the user positions the pointer over the

tool. You can use a text key or the literal text.

For example, **@language-toolbar-identify-point-tooltip** or **Find out about a location on the map**.

- **hideOnDisable:** If this property is set to `true` and the tool's command cannot run, the tool does not show in the toolbar.  
If `hideOnDisable` is `false` and the tool's command cannot run, the tool shows in the toolbar, but it is grayed out.
- **isSticky:** When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.  
If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
- **statusText:** The status message to display when the tool is activated, often containing instructions for the user. You can use a text key or the literal text.  
For example, **@language-toolbar-identify-point-desc** or **Click or tap a location on the map to learn what's there**.

## Properties of Regions

- **id:** A unique ID for this `region`.
- **type:** The type is `region`.
- **regionName:** A unique name for this `region`. This property is referred to by transient elements to create context-sensitive toolbars associated with an application state.


## Properties of Flyouts

- **id:** A unique ID for this `flyout`.
- **type:** The type is `flyout`.
- **name:** The name that you want to appear on the Multitool. You can use a text key or the literal text.  
For example, **@language-toolbar-markup-drawing-tools** or **Draw**.
- **items:** An array of items, each of which is either a button, toggle button, or tool.
- **layout:** This property is not used.
- **layout:** To display large icons for each tool, set to `Large`, otherwise, set to `Small`. The default is `Large`.

## Views

- **TabbedToolbarView:** None
- **TabbedToolbarButtonView:** None
- **ToolbarFlyoutView:** None
- **IWantToMenuButtonView:** (Handheld interface only) None

- **SearchView:** (Handheld interface only) None
- **NavBarSmallView:** (Handheld interface only) None
- **NavBarButtonView:** (Handheld interface only) None

 As of version 2.4, the `ToolBarView` and `ToolBarManagedViewsView` in previous versions of the Handheld interface no longer exist.

## View Models


- **TabbedToolBarViewModel:**
  - **toolbarGroupRefs:** An array of the IDs of the configured groups that you want to appear in the toolbar. This property provides a way to hide a group that is configured in the toolbar, without removing the configuration for the group. This is useful if you want to remove a group, but you think that you might want to add it back later.  
Hide the group by removing its ID from the `toolbarGroupRefs` list. Add the group back by adding its ID back to the `toolbarGroupRefs` list.
  - **toolbarOpenByDefault:** To open the toolbar when the viewer starts, set to `true`; otherwise, set to `false`. The default is `false`.
- **TabbedToolBarTransientViewModel:** None
- **NavBarButtonViewModel:** (Handheld interface only) None

### See Also...

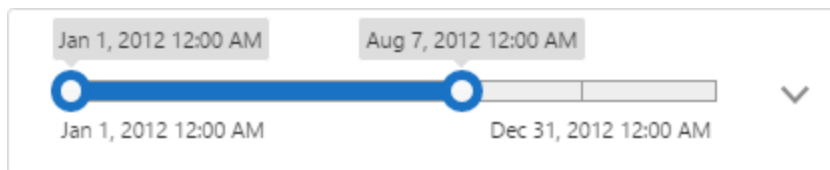
[About User Interface Text on page 64](#)

[CompactToolBar Module on page 150](#)

## 15.74 TimeSlider Module

 You can configure the time slider map widget in the viewer's Management Pack. See [Configure Map Widgets on page 90](#) for more information.

The TimeSlider Module allows users to interact with time-aware layers on the map. The time slider map widget can display changes to features on the map over time.



The time slider map widget (collapsed)

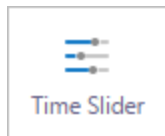


The time selected on the timeline corresponds with the features that are displayed on the map. If a feature's timestamp is outside of the selected time, it is hidden from users. Timelines display changes to all of the visible time-aware layers on the map at once.

You can change the selected time using the handles on the timeline. Depending on the configured Time Mode setting, there may only be one handle. For more information see [Time Modes](#).


## Enable Time Sliders from the Toolbar

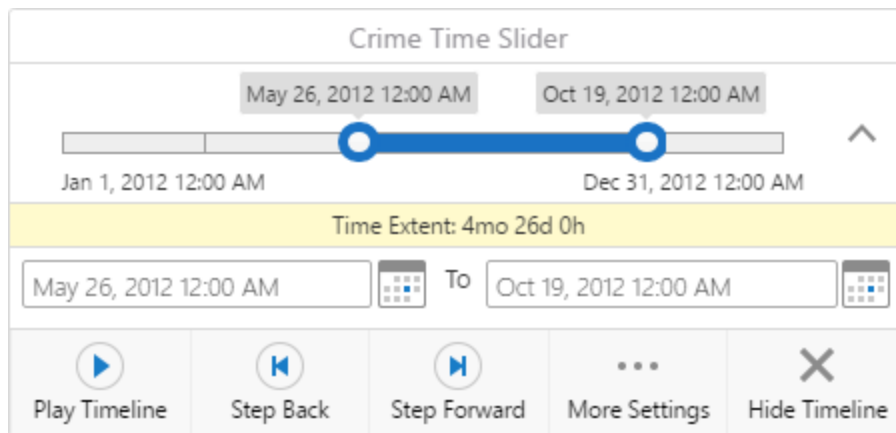
You can configure the Time Slider tool in the toolbar from the viewer's Management Pack. Find the **Time Slider** tool in the list of Available Tools and drag it to the Configured Toolbar pane. You must press Apply Changes and Save Site to apply your changes to the toolbar.



The Time Slider tool

## Expanded Time Slider

The time slider can be expanded using the  drop-down node to reveal additional settings. It also displays the name of the selected time slider profile and displays the full time extent amount in a banner. In the image below, the time extent is "4mo 26d 0h".



The time slider map widget (expanded)

The expanded time slider includes the Play Timeline, Step Back, and Step Forward controls. The controls allow users to animate the changes to visible time-aware layers over time. The selected position on the timeline moves forward or backward consecutively, showing or hiding features based on their time information.

The following functionality is available from the expanded time slider map widget:

- **Start Time:** Sets a start date and time for the timeline using the text input or calendar interface.
- **End Time:** Sets an end date and time for the timeline using the text input or calendar interface.
- **Play Timeline:** Animates the changes in the time-aware layer from the beginning to the end.

The selected time is highlighted.

- **Step Back:** Selects the previous position on the timeline.
- **Step Forward:** Selects the time extent's next position on the timeline.
- **More Settings:** Opens the Time Slider Settings panel.  
See [Time Slider Settings](#) for more information about the settings.
- **Close Timeline:** Closes the time slider map widget so it is no longer visible.

## Time Slider Settings

When you select the More Settings button on the expanded time slider widget, the Time Slider Settings panel opens. It allows users to further customize the time slider configuration:

- **Time Slider Profile:** Selects another configured time slider to use.  
You can create time slider profiles in Essentials Manager. For information, see "Time Sliders" in the *Essentials Administrator Guide*.
- **Time Mode:** Chooses the timeline's time mode from a drop-down menu.  
For more information about the available time modes, see [Time Modes](#).
- **Time Interval:** Chooses how many time interval units should be between each selectable position on the time line.  
For example, a time-aware layer has 12 months of changes and the Time Interval Unit is set to months. If the Time Interval is set to 1, then the timeline has twelve unique positions.
- **Time Interval Unit:** The unit of time that the timeline should display.



Each tick on a timeline represents an instant, as defined by the Time Interval and Time Interval Unit settings. The timeline shows a maximum of 50 instants. For example, even if the time interval is set to milliseconds and the time extent is set to 3 months, the timeline limits the selectable instants to 50. We recommend setting time intervals so that there are fewer than 50 instants on the timeline.

- **Playback Delay (Seconds):** When the Play Timeline button is selected, the Playback Delay setting slows down the animation. The delay is set in seconds.
- **Loop Playback:** If the checkbox is selected, playback is looped when the Play Timeline button is selected.

## Time Modes

The TimeSlider Module includes three time modes. Time modes change how the time extent can be selected.

- **Cumulative From Start:** Displays cumulative data over a time span. The slider's handle can be dragged to set the end of a time span. Unlike the Time Extent mode, there is no handle to set the beginning of the time span. The end user uses the handle to show more or less data as it accumulates over time.
- **Time Extent:** Displays a range of time that can be manipulated. The range is defined by two draggable handles which mark the beginning and end of the time extent.

- **Time Instant:** Displays a particular instant in time. If the time-aware layer does not have data for that particular instant in time, the map renders no data.

## Configuration Properties

### Module

- **behaviors:** An array of named behaviors that run when an associated event occurs. By default, the behaviors are:
  - **TimeSliderCloseTimelineActionInvokedBehavior:** A behavior that runs an array of commands when the user closes the timeline map widget. By default, the behavior runs three commands: `HideTimeSliderSettings`, `CloseDataFrame`, and `HideTimeSlider`.
  - **TimeSliderMoreSettingsActionInvokedBehavior:** A behavior that runs an array of commands when the user selects the More Options button on the timeline map widget. By default, the behavior runs two commands: `ToggleTimeSliderActions` and `ShowTimeSliderSettings`.

### Views

- **TimeSliderView:** None
- **TimeSliderSettingsView:** None

### View Models

- **TimeSliderViewModel:** Configure how the time slider map widget appears in the viewer.
  - **enabled:** Enables the time slider map widget. The default value is `true`.
  - **animateSlider:** Enables smoother time slider animations when playing a timeline or using the Step Back and Step Forward buttons. The default value is `true`.
  - **activateOnStartup:** Activates the time slider when the viewer is loaded. The default value is `true`.
  - **maxTimeExtentDisplayUnits:** Measures the time extent selected on the timeline and displays the full amount of the time extent. For example, "1yr 2mo 3d 30h". The default value is `3`, which means the viewer displays three time units ("1yr 2mo 3d").
  - **noOfflineSupportStatusMsgTimeoutSecs:** The time slider becomes unavailable if a viewer goes offline. In that case, a status message appears informing the user that the time slider is unavailable. This setting configures how long the message should remain on the screen in seconds. The default value is `10`.

## 15.75 Tools Module

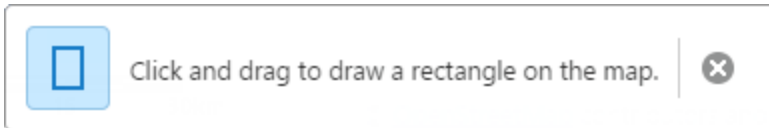
The Tools Module makes it possible to create tools for other modules.

Any module can configure and register a tool. You can configure a module to contain a `tools` node that defines the tools to create for that module. Modules that contain tools should use the Tool Registry object of the Application to register their tools.

## Configuration Properties

### Module

- **showStatusMessages:** When this property is set to `true`, a tool's `statusText` is displayed on the map when the user selects the tool. By default, `showStatusMessages` is `true`.  
Every tool has a `statusText` property. In the factory configuration, `statusText` properties are used to provide instructions for how to use the tool.  
For example, the status text for the Rectangle drawing tool is "Click and drag to draw a rectangle on the map."  
When the user selects the Rectangle drawing tool, the text appears on the map.



#### Status Text message for the Rectangle drawing tool displayed on the map

- **tools:** An array of tools with the properties listed below. By default, the `tools` array is empty in the Tools Module. Tools are configured in the module that uses them. For example, the Identify Module has tools that perform different types of identify operation. Similarly, the Editing Module has editing tools.
  - **name:** The name of the tool.



If your viewer is going to be available in more than one language, enter the text key that the tool name is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

- **command:** The command that the tool runs.  
For a list of commands, see the [Geocortex SDK for HTML5 API Reference](#).
- **drawMode:** The type of geometry the user draws, upon which the tool operates.
- **isSticky:** When set to `true`, the tool remains selected after the user has used it. To deselect the tool, the user must click the tool a second time, or click a different tool. This allows the user to use a tool repeatedly without having to reselect it each time.  
If you do not want a tool to remain selected after it is used, set `isSticky` to `false`.
- **iconUri:** The image that displays beside the tool.
- **statusText:** The status message to display when the tool's input method is via mouse, often containing instructions for the user. You can use a text key or the literal text.
- **keyboardStatusText:** The status message to display when the tool's input method is via keyboard, often containing keyboard shortcut hints for the user. You can use a text key or the literal text.

### Views

The Tools Module does not have any views.

### View Models

The Tools Module does not have any view models.

**See Also...**

[Editing Module](#) on page 160

[Identify Module](#) on page 193

[IWantToMenu Module](#) on page 215

[About User Interface Text](#) on page 64

[Geocortex SDK for HTML5 API Reference](#) on page 422

## 15.76 UploadData Module

The UploadData module implements the ability for users to upload files containing spatial or address data. Once uploaded, the data is served as a new feature layer. If the uploaded data is published to ArcGIS Online or Portal for ArcGIS, the data can be queried and filtered. The user can interact with the data as they could any other feature layer.

The user who uploaded the data is the only user who can view it, and when the user's session ends the data is removed from the map. In this case, the data is automatically deleted from the server in one hour. However, the following methods enable users to share data across sessions:

- [Save the data as part of a project](#)
- [Publish data to an ArcGIS Online or Portal for ArcGIS account](#)

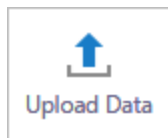
The UploadData Module is not available in offline mode.



The UploadData functionality is not available to iOS users because of file system security policies that limit access to files and directories.

### Add and Remove Uploaded Data

Data files with spatial or address data can be uploaded by users via the Upload Data tool. The Upload Data tool is configured from the Toolbars page of a viewer's Management Pack. See [Configure the Toolbar](#) on page 108 for instructions.



**The Upload Data tool.**

#### Add a User-Added Layer

A wizard guides users through the upload process. The dialog boxes appear in the following order:

1. The Add Data to Map upload form.  
The user browses their file system for files to upload.



The UploadData module only supports multiple simultaneous uploads for shapefiles. When a single shapefile is uploaded, the upload dialog will allow users to name the layer. When more than one shapefile is uploaded at once, the layers are named: <name>, <name> (0), <name> (1), and so on.

2. The Table Mapping Details dialog box.  
If the user uploaded a XLSX or CSV file, they must identify the uploaded data's geographical information. For more information, see [Table Mapping Details](#).
3. The Layer Details dialog box.  
The user creates a name for the new layer.
4. The Table Record Results dialog box.  
The viewer alerts the user whether each table record passed or failed.
5. The Symbolize Data dialog box.  
The user modifies how the new layer is represented on the map. The user can choose the color, size, and shape of the layer's symbols. For more information, see [Symbolize Data](#).

The user-added layer appears at the top of the layer list, unless you have specified that user-added layers should appear within a configured folder. For more information, see "The Layer List" in the *Essentials Administrator Guide*.

### Remove a User-Added Layer

To remove a user-added layer, press the **> Layer Actions** button next to the user-added layer you wish to remove. Then, use the **Remove Layer** action to remove the layer.

### Save Uploaded Data to Project

Projects allow users to save viewer sessions, share the sessions, and reload them later. A user can save uploaded data and load it in another session saving their session using the Projects panel. For more information about this feature, see [Project Module](#).

### Publish Data to ArcGIS Server or Portal for ArcGIS

The user can choose to publish the uploaded data to their ArcGIS Online or Portal for ArcGIS account if they are logged in. If the user has not been granted publishing privileges in ArcGIS Online or Portal, the user's data cannot be published. The data is published as a feature service. When the data is published, it is also stored in the user's content.

In order to publish, the URL to the user's ArcGIS account must be registered in the `proxy.config` whitelist:

```
<serverUrl url="https://organization.maps.arcgis.com/sharing/rest/"  
matchAll="true"></serverUrl>
```


Where `organization` is the configured ArcGIS Online organization name.

For information about publishing to ArcGIS, see [ExportWebMap Module](#).

## User-Added Layers in the Layer List

By default, user-added layers appear at the top of the layer list once added. From Manager, you can choose where your uploaded data appears in the Layer List.

### To Set Placement for User-Added Layers:

1. From the **Map** page in Manager, select the **Layer List** tab.
2. Select the  **Edit** icon next the folder where user-added layers should appear.
3. Select the  **Place user-added layers here** checkbox.
4. Press the **Apply Details** button and **Save Site**.

## Supported File Formats

Uploaded data files require either spatial coordinates or address data. While some supported file formats include spatial data by default, CSV and XLSX files require columns in their data tables specifically for spatial coordinates (**Latitude** and **Longitude**) or address data.

The UploadData module supports uploads in the following file formats:

- CSV
- XLSX
- KML
- GPX
- ZIP (for shapefiles and FileGDBs)

### CSV Uploads

The viewer accepts CSVs in UTF-8 encoding. It accepts other encodings if the file includes a [byte order mark](#). Both Microsoft Excel and Notepad can save CSV files in UTF-8 format.

### Shapefile Uploads

The following shapefile pieces are required to successfully publish shapefile data to ArcGIS Online or Portal for ArcGIS:

- .shp
- .shx
- .dbf
- .prj

If you do not plan to publish shapefile data, only the .shp file is necessary to generate shapefile data.



The UploadData module only supports multiple simultaneous uploads for shapefiles. When a single shapefile is uploaded, the upload dialog will allow users to name the layer. When more than one shapefile is uploaded at once, the layers are named: <name>, <name> (0), <name> (1), and so on.

## FileGDB Uploads

In order to upload a FileGDB file, the user must be logged in to an ArcGIS Online or Portal for ArcGIS account.

## GPX Uploads

GPX files cannot be published to ArcGIS Online or Portal for ArcGIS.

## File Size Limitations

The file size limit for uploaded files defaults to 20MB. You can configure a custom file size limit in the Essentials Web.config file:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\Default\REST
Elements\REST\Web.config
```

### To change the file size limitations for uploaded data:

1. Search for the string `maxAllowedContentLength`. It is embedded in the following line of XML:

```
<requestLimits maxAllowedContentLength="20971520" />
```

2. Configure the value of `maxAllowedContentLength` in bytes.
3. Search for the string `maxRequestLength`. It is embedded in the following line of XML:

```
<httpRuntime maxRequestLength="20480" requestValidationMode="2.0"
enableVersionHeader="false" />
```

4. Configure the value for `maxRequestLength` in kilobytes.  
This value should be equal to the value of `maxAllowedContentLength`.

## Table Mapping Details

The Table Mapping Details dialog box allows users to instruct Essentials how to parse uploaded data correctly. If the user uploaded a XLSX or CSV file, this dialog also displays a preview of the user's uploaded data for reference.

If the table's first row is a label, Essentials will attempt to detect the X axis and Y axis for the uploaded data.

If the table's first row is not a label or the table does not include spatial coordinates, the user can choose which columns to be used for feature locations and labels.



Table Mapping Details ✕

Spatial Coordinates     Address Data

X Axis Column:

Y Axis Column:

Spatial Reference:

Use first row as header

OBJECTID	NAME	latitude	longitude
1	Warner Brothers Ranch	34.15831573	-118.3449609
2	ABC Television Center	34.15753481	-118.2885937
3	Dreamworks	34.1572312	-118.2857111
5	ABC Television Center	34.10287537	-118.2809291

The Table Mapping Details dialog box

- **Spatial Coordinates:** If the uploaded file includes labelled columns with geographical coordinates (for example, columns labelled X and Y or Lat and Long), Essentials will detect this and select X Axis Column and Y Axis Column automatically.
  - By default, the Spatial Reference drop-down menu is set to WGS 1984. If you require another spatial reference system not available in the drop-down menu, you may select **Custom WKID/WKT** and enter a custom spatial reference in the text input box. To customize the available spatial references in this drop-down men, see the `sourceSpatialReferences` property in [Configuration Properties](#).
- **Address Data:** If the uploaded file does not include geographic coordinates, the user can assign columns with address information to be geocoded. A geocoding service must be configured in order to use this feature. For more information, see [Configure a Batch Geocoder](#).
- **Use first row as header:** Select this checkbox if the first row of your data table functions as a label for its corresponding column.



If your data table has no labels, you can label the spatial coordinates or address data using the corresponding drop-down menus. For example, if **Spatial Coordinates** is selected, you can set the X Axis Column and Y Axis Column.

## Autodetect Column Headers

When a user uploads data, Essentials needs to know which column contains the X Axis Coordinates, which column contains the Y Axis Coordinates, and so on. The software looks for commonly used headers in the data. For example, to find X Axis Coordinates, the software looks for columns called `Latitude`, `Lat`, or `x`. If it finds one, then the software gets the X coordinates from that column.

You can configure additional column headers for Essentials to look for using the `autoDetectionTerms` properties. For information, see [Configuration Properties](#).

## Configure a Batch Geocoder

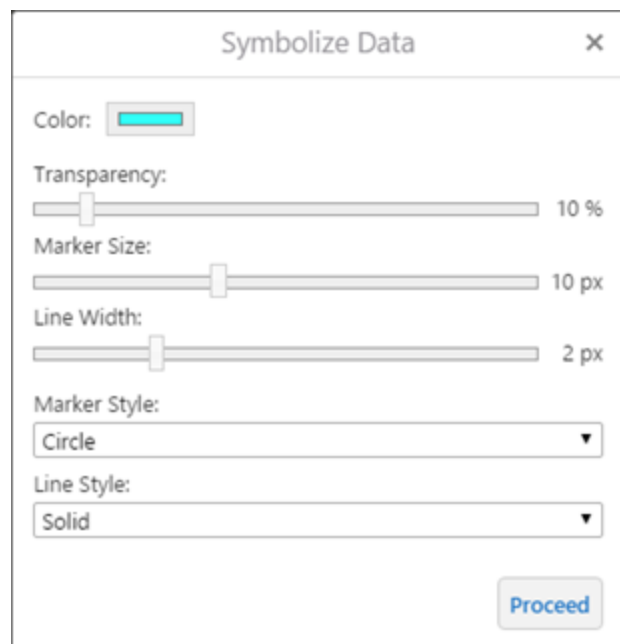
If uploaded CSV or XLSX data files do not provide spatial coordinates, address data needs to be geocoded using an ArcGIS Online geocoding service. To learn about batch geocoder configuration, see the section in the *Essential Administrator Guide* called Geocoding Services.

If none of your enabled geocoding services have the **Default Batch Geocoder** checkbox selected in their settings, Essentials cycles through your enabled geocoding services until it finds the first ArcGIS Online service listed.

All the parameters in your geocoding service's configuration are selectable in the Table Mapping Details dialog. Once the user has selected columns to match parameters, the value is used to batch geocode the uploaded data. If the user does not select a column to match with a parameter, a parameter's default value is used. For information on configuring a parameter's default value, see the section in the *Essential Administrator Guide* called Geocoding Services.

## Symbolize Data

Users can symbolize layers created from uploaded data files. Symbols are chosen during the during the file upload process. If an uploaded file includes multiple layers, like a FileGDB or KML, then symbols can be chosen for each layer.



The dialog box titled "Symbolize Data" contains the following settings:

- Color: Cyan
- Transparency: 10 %
- Marker Size: 10 px
- Line Width: 2 px
- Marker Style: Circle
- Line Style: Solid

A "Proceed" button is located at the bottom right of the dialog.

The Symbolize Data dialog box

End users can re-symbolize user-added layers on the map. To re-symbolize the layer, select the > Layer Actions button next to the layer in the layer list and choose **Turn on/off layer visualizations**. Then, you can use the Custom Layer Styles... option to re-symbolize the layer.

For more information about symbolization in the viewer, see [LayerStyles Module on page 235](#).

## Data Visualization Options

Users can visualize user-added feature layers with multiple point features as heat maps or clusters. For detailed information about heat maps and feature clustering, see [HeatMaps Module on page 189](#) and [ClusterLayers Module on page 148](#).

To turn on visualization, use the > Layer Actions button next to a user-added layer, then select **Turn on/off layer visualizations** and choose either Heat Map or Feature Clustering from the drop-down menu. When visualization is turned on, the user is presented with some visualization options.

The visualization options for heat maps include:

- **Intensity:** The size in pixels of the heat maps.
- **Intensity Multiplier:** Multiplies the intensity level by a selected numerical field from uploaded data.

The visualization options for feature clustering includes:

- **Radius:** The radius in pixels of each cluster label.
- **Maximum Cluster Size:** The maximum number of features represented by a cluster.

## Configuration Properties

### Module

- **checkJobStatusPeriod:** The period of time in milliseconds that the Viewer waits before re-checking if a file has been uploaded successfully. The default value is `4000`.
- **heatmap:** Visualizes uploaded data as a heat map.
  - **enabled:** A boolean that defines whether user-added layers have heat maps enabled initially. The default value is `false`.
  - **userCanToggle:** A boolean that defines whether users can toggle heat maps on user-added layers. The default value is `true`.
  - **respectScaleRange:** A boolean that defines whether a user-added layer's scale range is respected when heat maps are enabled. The default value is `true`.
  - **gradient:** Definition for the heat map color gradient as an array of four RGBA values. The order of the array represents the order of the colors: from outermost, to outer, to inner, to innermost. The default values are `[255,255,255,0]`, `[0,0,255,98]`, `[255,0,0,139]`, `[255,255,0,181]`.
  - **offset:** An array of ratios used as color stops for user-added layer heat maps.
  - **intensity:** The default intensity to set the heat map renderer in pixels. The initial value is `25`.
  - **includeInLegend:** A boolean that defines whether heat map colors appear in the map legend. The default value is `true`.

- **cluster:** Visualizes uploaded data as clusters on the map.
  - **enabled:** A boolean that defines whether user-added layers have heat maps enabled initially. The default value is `false`.
  - **userCanToggle:** A boolean that defines whether users can toggle clusters on user-added layers. The default value is `true`.
  - **radius:** The radius of a cluster in pixels. The default value is `50`.
  - **maximumFeatures:** The maximum number of features that can make a cluster. The default value is `100`.
  - **backgroundColor:** The background color of a cluster in an RGB color format. The default value is `[0, 0, 255]`.
  - **labelColor:** The color of a cluster's label text in an RGB color format. The default value is `[255, 255, 255]`.

## Views

- **UploadDialogView:** None
- **TableMappingDialogView:** None
- **LayerDetailsDialogView:** None
- **TableRecordResultsDialogView:** None
- **SymbolDialogView:** None

## View Models

- **UploadDialogViewModel:** None

### **TableMappingDialogViewModel:**

- **autoDetectionTerms:** Sets additional column headers for Essentials to look for. If a configured column header is detected, it is matched with a standard feature detail. For example, a column header `Movie Theater` could set the `featureLabel` attribute if configured to do so. Detection can be configured for the following attributes:
  - **xAxis:** An array of used to detect X-axis coordinates from a column header. The default value is `["Latitude", "Lat", "x"]`.
  - **yAxis:** An array of objects used to detect Y-axis coordinates from a column header. The default value is `["Longitude", "Long", "y"]`.
  - **zAxis:** An array of objects used to detect Z-axis coordinates from a column header. The default value is `["Altitude", "Alt", "z"]`.
  - **featureLabel:** An array of objects used to detect feature labels from a column header. The default value is `["Name", "Label", "Title"]`.
  - **Address:** An array of objects used to detect street addresses from a column header. The default value is `["Street", "Address"]`.
  - **Neighborhood:** An array of objects used to detect neighborhoods from a column header. The default value is `["Neighborhood"]`.
  - **City:** An array of objects used to detect cities from a column header. The default value is `["City"]`.

- **Subregion:** An array of objects used to detect subregions from a column header. The default value is ["Subregion"].
  - **Region:** An array of objects used to detect regions from a column header. The default value is ["Region", "State", "Province"].
  - **Postal:** An array of objects used to detect postal codes from a column header. The default value is ["Postal", "Zip"].
  - **PostalExt:** An array of objects used to detect postal extensions from a column header. The default value is ["postal\_ext", "postal\_ext"].
  - **CountryCode:** An array of objects used to detect country codes from a column header. The default value is ["countrycode", "country\_code", "cc"].
  - **SearchExtent:** An array of objects used to detect search extents from a column header. The default value is ["searchextent", "search\_extent"].
- **sourceSpatialReferences:** Specifies which Esri spatial reference WKIDs (well-known IDs) to display as options during the data upload process. The default preselected value is WGS 1984 (WKID 4326). Each spatial reference requires the following:
    - **label:** The label for the spatial reference in the dialog.
    - **spatialReference:** An object that requires a Esri spatial reference WKID. For a list of available WKIDs, see [the ArcGIS Geographics Coordinate Systems list](#).
      - **wkid:** The well-known ID number. For example, 4326.
    - **preselected:** A boolean that sets one Esri spatial reference WKID as the preselected option in the dialog. The default preselected value is WGS 1984 (WKID 4326).
  - **LayerDetailsDialogViewModel:** None
  - **TableRecordResultsDialogViewModel:** None
  - **SymbolDialogViewModel:** None

## Widgets

- **AttributeSymbologySettings:** Configures the default behavior for Attribute Symbology. This type of symbology allows users to style features in a layer based on the information in a specific field. For example, if a layer included a field with elevation data, the symbology could be rendered with up to twelve different colors, each color representing an elevation range.
  - **maxRenderClasses:** Sets the maximum number of classes that can be distinguished for the configured attribute. For example, you may wish to configure three classes for three different tree species that your layer identifies. The default value is 12. Note that increasing the default to a value greater than 12 can impact performance.
  - **maxSamples:** Defines how many features should be sampled to determine the values of a layer's render classes. The default value is 1000.
  - **defaultPointColor:** An array that defines the default color for point features in RGBA format. The default value is [150, 150, 150, 0.8].
  - **defaultPointSize:** The default size of a point feature in pixels. The default value is 12.

- **defaultLineColor**: An array that defines the default color for line features in RGBA format. The default value is [75, 75, 75, 1].
- **defaultLineWidth**: The default width of a line feature in pixels. The default value is 2.
- **defaultFillColor**: The default fill color for features in RGBA format. The default value is [150, 150, 150, 0.3].
- **defaultSymbologySettingsConfig**:
  - **selectOutlineColor**: Allows the user to select an outline color. The default value is `true`.
  - **alwaysUseColorSwatches**: Allows the user to always choose colors using color pickers. The default value is `false`.
  - **numberOfColorSwatches**: Defines how many color pickers should be available from the attribute symbology settings. The default value is 6.
  - **transparency**: Sets the user configuration options for the transparency slider. These properties accept values between 0 and 100 percent, where 0 is opaque and 100 is transparent.
    - **min**: Sets the minimum transparency value available, in percent. The default value is 0.
    - **max**: Sets the maximum transparency value available, in percent. The default value is 90.
    - **value**: Sets the initial transparency value, in percent. The default value is 10.
    - **step**: Sets the increment to use when the user increases or decreases the transparency value, in percent. For example, if the value is 10, there are 10 unique transparency values that can be picked between 0 and 100. The default value is 5.
  - **lineWidth**: Sets the user configuration options for the symbol line width slider.
    - **min**: Sets the minimum configurable width of the line, in pixels. The default value is 0.
    - **max**: Sets the maximum configurable width of the line, in pixels. The default value is 5.
    - **value**: Sets the initial configured line width value, in pixels. The default value is 2.
    - **step**: Sets the increment to use when the user increases or decreases the line width, in pixels. The default value is 1.
  - **markerSize**: Sets the user configuration options for the marker size slider.
    - **min**: Sets the minimum configurable marker size, in pixels. The default value is 1.
    - **max**: Sets the maximum configurable marker size, in pixels. The default value is 50.
    - **value**: Sets the initial configured marker size value, in pixels. The default value is 16.
    - **step**: Sets the increment to use when the user increases or decreases the marker size, in pixels. The default value is 1.
  - **markerStyles**: An array of objects that define available marker styles. By default, the styles included are `circle`, `diamond`, `cross`, `x`, and `square`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).

- **style**: A string that references an available marker style.
- **label**: The label to display for the marker style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for marker style language keys are `@language-symbology-settings-marker-style-*`, where `*` is the marker style—`circle`, `diamond`, and so on.
- **lineStyles**: An array of objects that define available line styles. By default, the styles included are `solid`, `dash`, `dot`, and `dashdot`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style**: A string that references an available line style.
  - **label**: The label to display for the line style in the viewer. The line style is the style of the border of the feature marker. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for line style language keys are `@language-symbology-settings-line-style-*`, where `*` is the marker style—`solid`, `dash`, and so on.
- **fillStyles**: An array of objects that define available fill styles. The fill is the most dominant pattern of the feature marker. By default, the styles included are `solid`, `forwarddiagonal`, `backwarddiagonal`, `cross`, `horizontal`, and `vertical`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style**: A string that references an available fill style.
  - **label**: The label to display for the fill style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for fill style language keys are `@language-symbology-settings-fill-style-*`, where `*` is the marker style—`solid`, `forward-diagonal`, `backward-diagonal`, and so on.
- **SymbologySettings**: Configures the default behavior for Simple Symbology.
  - **alwaysUseColorSwatches**: Defines whether color pickers should always be available. The default value is `false`.
  - **numberOfColorSwatches**: Defines how many color pickers should be available from the attribute symbology settings. The default value is `6`.
  - **transparency**: Sets the user configuration options for the transparency slider. These properties accept values between 0 and 100 percent, where `0` is opaque and `100` is transparent.
    - **min**: Sets the minimum transparency value available, in percent. The default value is `0`.
    - **max**: Sets the maximum transparency value available, in percent. The default value is `90`.
    - **value**: Sets the initial transparency value, in percent. The default value is `10`.
    - **step**: Sets the increment to use when the user increases or decreases the transparency value, in percent. For example, if the value is `10`, there are 10 unique transparency values that can be picked between 0 and 100. The default value is `5`.

- **lineWidth:** Sets the user configuration options for the symbol line width slider.
  - **min:** Sets the minimum configurable width of the line, in pixels. The default value is 0.
  - **max:** Sets the maximum configurable width of the line, in pixels. The default value is 5.
  - **value:** Sets the initial configured line width value, in pixels. The default value is 2.
  - **step:** Sets the increment to use when the user increases or decreases the line width, in pixels. The default value is 1.
- **markerSize:** Sets the user configuration options for the marker size slider.
  - **min:** Sets the minimum configurable marker size, in pixels. The default value is 1.
  - **max:** Sets the maximum configurable marker size, in pixels. The default value is 50.
  - **value:** Sets the initial configured marker size value, in pixels. The default value is 16.
  - **step:** Sets the increment to use when the user increases or decreases the marker size, in pixels. The default value is 1.
- **markerStyles:** An array of objects that define available marker styles. By default, the styles included are `circle`, `diamond`, `cross`, `x`, and `square`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style:** A string that references an available marker style.
  - **label:** The label to display for the marker style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for marker style language keys are `@language-symbolology-settings-marker-style-*`, where `*` is the marker style —`circle`, `diamond`, and so on.
- **lineStyles:** An array of objects that define available line styles. By default, the styles included are `solid`, `dash`, `dot`, and `dashdot`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style:** A string that references an available line style.
  - **label:** The label to display for the line style in the viewer. The line style is the style of the border of the feature marker. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default values for line style language keys are `@language-symbolology-settings-line-style-*`, where `*` is the marker style —`solid`, `dash`, and so on.
- **fillStyles:** An array of objects that define available fill styles. The fill is the most dominant pattern of the feature marker. By default, the styles included are `solid`, `forwarddiagonal`, `backwarddiagonal`, `cross`, `horizontal`, and `vertical`. You can configure additional styles that exist in the [ArcGIS API for JavaScript](#).
  - **style:** A string that references an available fill style.
  - **label:** The label to display for the fill style in the viewer. You can use text or a language key. For more information on using language keys, see [About User Interface Text on page 64](#). The default



values for fill style language keys are @language-symbology-settings-fill-style-\*, where \* is the marker style —solid, forward-diagonal, backward-diagonal, and so on.

## 15.77 User Module

The User Module implements user sign-in and sign-out to sites that are secured using Essentials Security.

Depending on your security configuration, users may be prompted to sign in when they launch a viewer. For information about Essentials Security, refer to the *Geocortex Essentials Administrator Guide*.

In addition, the HTML5 Viewer has links and tools that users can use at any time to sign in or sign out:

- **Hyperlinks in the Banner:** In the Desktop and Tablet interfaces, the viewer's banner has links that the user can click to sign in or sign out. You can hide one or both of the links using the settings on Manager's Permissions page. The Handheld interface does not have sign-in and sign-out links in the banner.
- **Menu Items in the I Want To Menu:** The Handheld interface has sign-in and sign-out options in the I Want To menu. If you want, you can remove one or both of these options. See [IWantToMenu Module on page 215](#) for information.
- **Tools in the I Want To Menu:** The Global menu, which is contained within the I Want To menu, has tools for signing in and out. The Desktop, Tablet, and Handheld interfaces all show the Global menu by default. You can remove one or both tools from the Global menu if you want. See [GlobalMenu Module on page 187](#) for information.

If you prefer to customize how users sign in and sign out, you can create hyperlinks, toolbar tools, menu items, and workflows that invoke the User Module's `SignIn` and `SignOut` commands.

The Desktop and Tablet interfaces have two views and two view models. `SignInView` and `SignInViewModel` control sign-in. `UserInfoView` and `UserInfoViewModel` control sign-out. By default, the sign-in and sign-out hyperlinks are in the viewer's `BannerContentRegion`. The Handheld interface does not have any views or view models.

## Configuration Properties

### Module

None



### Views

None



As of HTML5 Viewer 2.3, `UserInfoView` and `SignInView` were moved from the User Module to the [Banner Module](#) in both the Desktop and Tablet interfaces.

## View Models

- **SignInViewModel:**
  - **linkColor:** The color to use for the sign-in hyperlink's text. The color can be an HTML color name or a hexadecimal color, for example, **green** or **#00FF00**. The default color is **#1B7DBF** (blue). If the default color does not show up well against the background color, change `linkColor` to a color that contrasts better.
- **UserInfoViewModel:**
  - **linkColor:** The color to use for the sign-out hyperlink's text. The color can be an HTML color name or a hexadecimal color, for example, **green** or **#00FF00**. The default color is **#1B7DBF** (blue). If the default color does not show up well against the background color, change `linkColor` to a color that contrasts better.
  - **textColor:** The color to use for the text in the panel that opens when the user clicks the down arrow  beside the sign-out hyperlink. The color can be an HTML color name or a hexadecimal color, for example, **green** or **#00FF00**. The default color is **#333333** (gray). Configure `textColor` and `backgroundColor` so the panel's text is easy to read.
  - **backgroundColor:** The color to use for the background of the panel that opens when the user clicks the down arrow  beside the sign-out hyperlink. The color can be an HTML color name or a hexadecimal color, for example, **green** or **#00FF00**. The default color is **#FFFFFF** (white). Configure `textColor` and `backgroundColor` so the panel's text is easy to read.

### See also...

[GlobalMenu Module on page 187](#)

[Geocortex SDK for HTML5 API Reference on page 422](#)

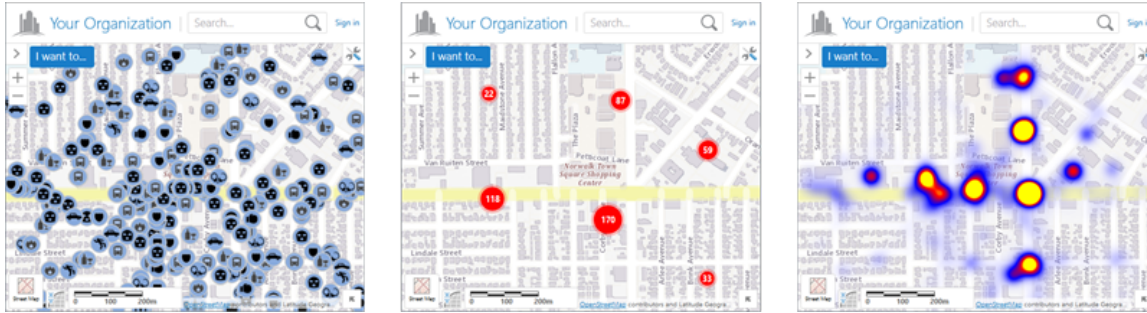
[Configure the I Want To Menu on page 79](#)

## 15.78 Visualization Module

The Visualization Module is a container that hosts visualization providers. There are several ways to visualize layers:

- Default symbology
- Clustering
- Heat maps
- Administrator-defined styles
- User-defined styles.

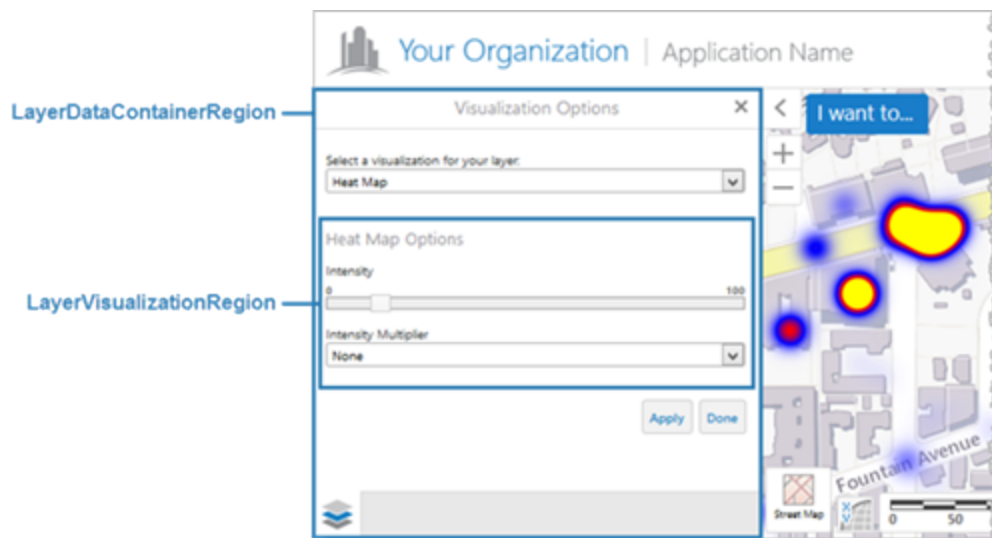
For additional information, see [LayerStyles Module on page 235](#), [ClusterLayers Module on page 148](#), and [HeatMaps Module on page 189](#).



Features shown individually (left), clustered (center), and in a heat map (right)

The Visualization Module also implements the Visualization Options panel, where users can change the current visualization and its settings. Activating the Visualization Module's `VisualizationView` opens the Visualization Options panel.

By default, `VisualizationView` is hosted in `LayerDataContainerRegion`. `VisualizationView` contains `LayerVisualizationRegion`, which hosts the settings for the currently selected visualization.



Regions used for the Visualization Options panel

The Visualization Module provides two commands and two events, which you can use in hyperlinks and workflows:

- The `ShowVisualizationView` command opens the Visualization Options panel for the specified feature layer, provided the feature layer has one or more visualizations configured for it. Note that if a map Service ID is used with `ShowVisualizationView`, then the Visualization Options panel only applies to the first layer from the service.
  - `VisualizationViewActivatedEvent` is raised when the Visualization Options panel opens.
  - `LayerVisualizationChangedEvent` is raised when the type of visualization changes, for example, when the user selects a different visualization from the Visualization Options panel.
- The `RemoveVisualization` command removes the current visualization from the specified feature layer, if there is an active visualization. At most one visualization can be active at a time.

For more information about commands and events, refer to the Geocortex SDK for HTML5 API Reference. Instructions for accessing the API Reference are [here](#).

## Configuration Properties

### Module

None

### Views

- **VisualizationView:** None

### View Models

- **VisualizationViewModel:**
  - **containerRegionName:** The name of the region to display within `VisualizationView`'s region. The default is `LayerVisualizationRegion`, which is used by the `ClusterLayers` Module and the `HeatMaps` Module to display their user-configurable settings.
  - **defaultDisplayName:** The text for the visualization option that means "do not show a visualization". The default text is **None**. Visualization options appear in the drop-down list on the Visualization Options panel.
  - **containerTitle:** The title that appears at the top of the Visualization Options panel. The default language string, `@language-visualization-title`, is set to **Visualization Options** by default.
  - **visualizationProviders:** An array of visualization providers. Each provider provides a way to visualize features on the map. By default, there are three providers, one for layer styles, one for heat maps, one for clustering. Each visualization provider has the following properties:
    - **type:** The provider type.
    - **viewID:** The ID of the view to show in the container, `containerRegionName`.
    - **displayName:** The text for this provider's visualization option, which appears in the drop-down list on the Visualization Options panel.
    - **libraryId:** (optional) The ID of the library containing your custom-developed visualization providers. If you are using the built-in visualization providers only, omit the `libraryId` property.

### See Also...

[LayerStyles Module on page 235](#)

[ClusterLayers Module on page 148](#)

[HeatMaps Module on page 189](#)

## 15.79 Workflow Module

The Workflow Module renders workflows.

There are two steps to configure a viewer to use a workflow:

- Configure a method to run the workflow in the viewer.
- Configure workflow containers to manage the workflow's presentation in the viewer.

These are the viewer steps only. You must also create the workflow in Workflow Designer and add the workflow to the site that the viewer belongs to. For more information, see "Overview of Steps to Create and Use a Workflow" in the *Geocortex Essentials Administrator Guide*.

There are several ways to configure a viewer to allow end users to run a workflow—add a [tool](#), [menu item](#), or hyperlink that runs the workflow. These methods use one of the viewer's workflow commands to run the workflow. For a list of workflow commands, see the [Geocortex SDK for HTML5 API Reference](#).

In addition, you can configure a workflow to run automatically whenever the viewer launches. For instructions, see "Configure a Workflow to Run On Startup" in the *Geocortex Essentials Administrator Guide*.

## Configure Workflow Containers

Workflow containers define the viewer [regions](#) where workflows display content to the user. If a workflow does not display anything in the viewer, then you do not need to configure workflow containers.

Workflow containers are defined in the viewer's Workflow Module and referenced in the workflow. When you create a workflow container in a viewer, you assign a name to the container and specify the region where you want the viewer to display the content. You can also configure a workflow container's title, icon, and other settings that affect the presentation of workflow content.

You use the container's name to reference the container in the workflow. For instructions, see "Configure Workflow Containers" in the Workflow Designer help system.

## Configuration Properties



Starting in version 2.4 of the HTML5 Viewer, the `startupWorkflows` module property is deprecated. Startup workflows are now configured in the site. For information, see "Configure a Workflow to Run on Startup" in the *Geocortex Essentials Administrator Guide*. Startup workflows that were configured using `startupWorkflows` will continue to work in version 2.4 and higher.

## Module

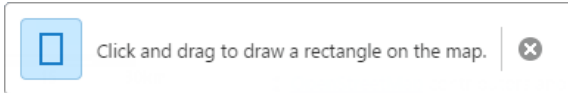
- **showTitleInFormBody:** When this property is set to `true`, the title appears in the body of the form along with the header.
- **defaultContainerRegionName:** The name of the region in which the workflow is run. The default is `DataRegion`.
- **defaultContainerTitle:** The default title of the container hosting the workflow.



If your viewer is going to be available in more than one language, enter the text key that the title is assigned to. See [About User Interface Text on page 64](#) for more information on using text keys.

The default is `@language-workflow-title`.

- **defaultContainerIconUri:** The default image to be displayed with the title of the container hosting the workflow.
- **showCaptureStatusMessages:** When set to `true`, the capture geometry messages are shown in the status module. The default is `true`.



#### Example of a capture message displayed on the map

- **displayResultPickerTemplateComplexity:** The amount of information to display in query results from a workflow that uses the workflow Display Result Picker activity. The default for the Desktop interface is `complex`. For the Tablet and Handheld interfaces, the default is `simple`.
- **icons:** A list of icon URLs followed by associated replacement icon URLs. For example, given an icon URL, you might want to replace it with a certain icon URL for the Desktop interface, but a different icon URL for the Handheld interface. You may add as many icon URL and replacement icon URL pairs as you want.
- **containers:** An array of workflow containers with the following properties:
  - **name:** The name of the container.
  - **title:** The title of the workflow. You can use a text key or the literal text.
  - **regionName:** The name of the region to host the container.
  - **iconUri:** The URI of the image to be displayed with the title.
  - **allowClose:** To display a close button for the container, set to `true`; otherwise, set to `false`.

#### Views

- **WorkflowListView:**
  - **hideOnClickWorkflow:** When this property set to `true`, the workflow list is hidden when a workflow from the list is selected by the user.

#### View Models

- **WorkflowViewModel:** None

#### See Also...

[About User Interface Text on page 64](#)

[Status Module on page 363](#)

[Tools Module on page 375](#)

## 15.80 WorkflowHost Module

The WorkflowHost module allows the Geocortex Workflow standalone product to work in the Geocortex Viewer for HTML5. For more information about the standalone product, see the Geocortex Workflow information at <https://docs.geocortex.com/workflow>.

For information about the original implementation of Workflow in Essentials see:

- The "Workflows" topic in the *Geocortex Essentials Administrator Guide*.
- [Workflow Module on page 392](#)

## Configuration Properties

Module

None

Views

None

ViewModels

None

## 15.81 ZoomControl Module

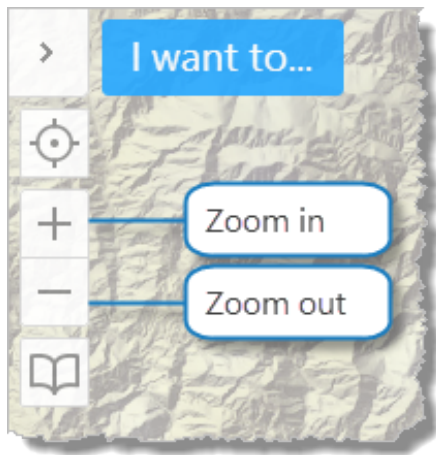
The ZoomControl Module is responsible for map zoom controls.



The views are configured in the [Navigation Module](#), which determine the order in which all navigation-related views appear.



`GeolocateViewModel` has been moved to the Geolocate Module.



### Zoom controls on the map

- **Zoom in:** Zooms in the map.
- **Zoom out:** Zooms out the map.

## Configuration Properties

### Module

None

### Views

None

### View Models

None

### See Also...

[Navigation Module on page 285](#)

## 16 Viewer Commands

### 16.1 About Commands

A viewer command is an instruction to the viewer to perform a particular action. For example, the `ShowLayerList` command tells the viewer to display the layer list.

The HTML5 viewer uses the commands in its repertory to implement the features and functions it offers. In addition, you can use viewer commands in hyperlinks and workflows, as described below.

Because commands typically perform a very simple action, the HTML5 viewer sometimes runs several commands one after the other to perform more complex actions.

Some commands need objects or values to work with when performing their action—these objects and values are the command's parameters. The definition of a command specifies how many parameters the command has, the type of each parameter, and whether the parameter is required or optional. For example, the `Alert` command has two required parameters—the alert's title and message, both strings—and one optional parameter—a function that indicates how the user responded to the alert.

#### 16.1.1 Viewer Commands in Hyperlinks

Many viewer commands can be embedded in HTML anchor elements that are configured in feature descriptions. When the user clicks the hyperlink in the feature description, the command executes.

For information on configuring viewer commands in hyperlinks, refer to the "Viewer Commands in Hyperlinks" section in the *Geocortex Essentials Administrator Guide*.



► **To determine if a particular command can be used in hyperlinks:**

1. Find the command in the [Geocortex SDK for HTML5 API Reference](#).
2. Look in the command's **Description** column.  
Commands that cannot be used in hyperlinks have a note in the description column. If there is no such note, then you can safely use the command in hyperlinks.

## 16.1.2 Viewer Commands in Workflows

The Run External Command workflow activity runs the specified viewer command. This means you can create workflows that use many of the same commands that the viewer itself uses.

For information on the Run External Command activity, refer to the "Activity Library" | "Common Viewer Activities" | "Run External Command" section of the Workflow Designer help system (run Workflow Designer from the Start menu, and then click the help icon in the toolbar).

► **To determine if a particular command can be used in workflows:**

1. Find the command in the [Geocortex SDK for HTML5 API Reference](#).
2. Look in the command's **Description** column.  
Commands that cannot be used in workflows have a note in the description column. If there is no such note, then you can safely use the command in workflows. Some commands have restrictions on how they are used—these restrictions are also described.

## 17 Editing

End users can add and modify map features and related features in ArcGIS Tables. As an administrator, you control the user's editing experience through various modules and settings.

### Enable Editing on an ArcGIS Feature Layer

Editing features from an HTML5 viewer is controlled by the Editing Module. In order for features to be editable, they must belong to a feature layer with editing enabled from ArcGIS Server. For information on setting edit privileges on feature layers, see ArcGIS's documentation: [Allow others to alter hosted feature layers](#).

Once editing is turned on in ArcGIS Online, users can perform the following functions in a viewer:


- Add new features to a layer
- Edit feature details
- Edit a feature's geometry
- Remove a feature

To edit related features from ArcGIS Tables, you must also add the tables in Manager. For more information, see "ArcGIS Tables" in the *Geocortex Essentials Administrator Guide*.

## Layer Editing Configuration in Essentials Manager

From Manager, you can configure whether users are allowed to snap to features and whether snapping is the default behavior.

### ► To configure snapping for editable layers:


1. Go to the **Map** page of Manager.
2. Click the **Layer List** tab.
3. Click the  **Edit** icon next to the layer you want to configure.
4. On the Details tab, locate the Editing section.
5. Select or clear the following checkboxes:
  - **Allow Snapping:** To allow users to snap to features, select this checkbox.
  - **Snapping Enabled:** To enable snapping by default for features in this layer, select this checkbox.

The Features Copyable to Edit Layer checkbox is not configurable for HTML5 viewers.

6. Click **Apply Changes**.
7. Click **Save Site**.

## Offline Experience

Through the Geocortex Mobile App Framework, the Geocortex Viewer for HTML5 allows users to work in environments where connection to the Internet is sporadic or absent. Field workers and remote employees can venture into low-connectivity or no-connectivity areas, and continue to browse and edit spatial and non-spatial data—their normal GIS workflow is not interrupted.


A typical usage scenario would be a field worker who needs to use the Geocortex Mobile App Framework in a remote location lacking network connectivity. Before departing, the worker would download a suitable Offline Map. At the remote location, the worker would activate the Offline Map to continue working with the map, browsing and editing features. Upon returning to a place with network connectivity, the worker can use the Offline Map's  **Sync** button to synchronize any changes to the map with the server.

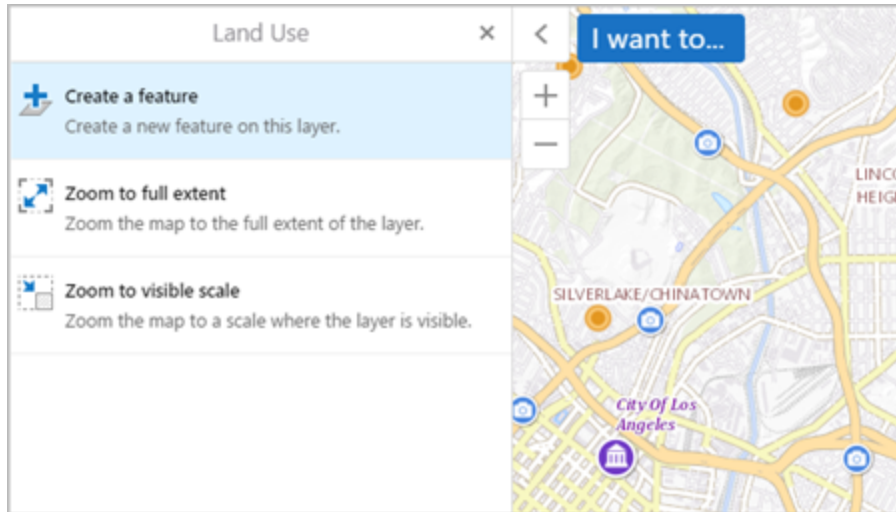
For more information see [Offline on page 401](#) and [Offline Module on page 287](#).

## User Interface

Users can create, modify, and remove features directly on the map. The viewer provides various user interface elements that initiate edit commands. While the following overview of user interface elements is not exhaustive, it offers an introduction to the scope of editing functionality.

### Create New Feature

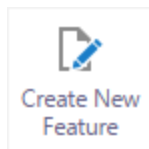
To add a new feature to a layer, users can locate the layer they want to add to in the Layer List and select the  **Layer Actions** button. If a layer is editable, the **Create New Feature** action is at the top of the available layer actions.



You can also configure your toolbar with the Create New Feature tool. When the user selects the Create New Feature tool, they are presented with a list of editable layers that they can create a feature for.

### Create New Feature Tool

Users can add new features with the Create New Feature tool. Ensure both Create New Feature and CreateNewFeatureControlRegion are included in your configured toolbar. When the user selects the Create New Feature tool, they are presented with a list of editable layers that they can create a feature for.



### The Create New Feature tool

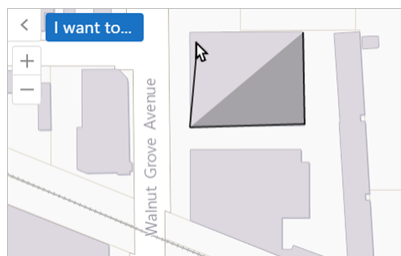


The toolbar can be configured using Manager. See [Configure the Toolbar on page 108](#) for instructions.

For more information see [Editing Module on page 160](#).

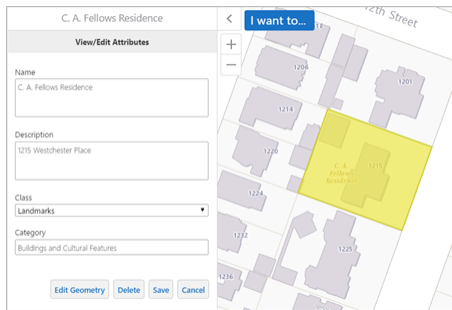
### Add Geometry

During the feature creation process, the user can add the feature's geometry to the map. The type of geometry depends on the feature layer and could be a point, line, or polygon.



### A user adds geometry to the map.

Once geometry is added, the user can add any feature attributes provided by the selected layer.

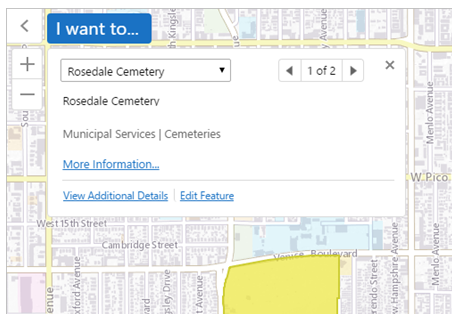


### The View/Edit Attributes panel.

Once a feature is ready, the user selects the **Save** button to send changes to the server.

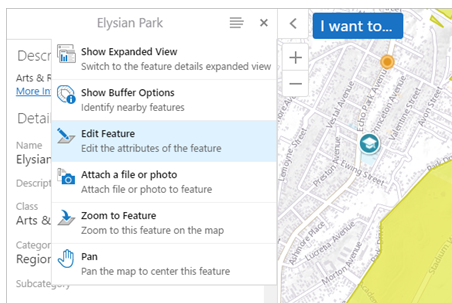
### Edit Feature

If a feature already exists, the user can press the **Edit Feature** link on the feature's map tip.



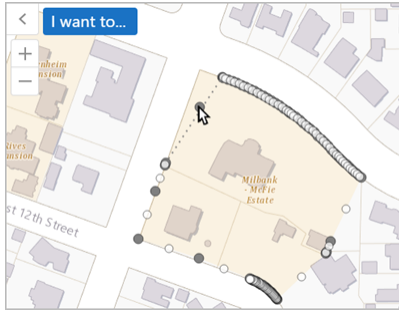
### The map tip for an editable layer. Notice the Edit Feature link.

Alternatively, the **Edit Feature** action appears in the list of available actions in the feature details view. To see the list of available actions, users can activate the panel actions menu by selecting the **Panel Actions** button.



### The Edit Feature action in a list of available actions.

The user can also edit a feature's geometry by selecting the **Edit Geometry** button while editing a feature. This activates a geometry editing tools on the map which you can use to adjust existing geometry. The type of geometry depends on the feature layer and could be a point, line, or polygon.




A user edits existing polygon geometry.

Once a feature has been edited, the user selects the **Save** button to send changes to the server.



### Delete Feature

To delete a feature, the user can use the **Edit Feature** link, scroll to the bottom of the Edit/View Attributes view, and use the **Delete** button. This removes the feature from the map as well as from the server.

### Create New Related Feature

To create a new related feature in an ArcGIS Table, while viewing the feature for which you want to create a related feature, click  **Create A New Related Feature**. Fill in the desired attributes, and click **Save**.

### Edit Related Feature

To edit a related feature from an ArcGIS Table, while viewing the feature that contains the related feature you want to edit, click the related feature, click the  **Panel Actions** menu, and select  **Edit Feature**. Modify the desired attributes, and click **Save**.

### See Also...

[Offline on page 401](#)

[FeatureLayer Module on page 176](#)


[Editing Module on page 160](#)

[Offline Module on page 287](#)


## 18 Offline


Through the Geocortex Mobile App Framework, the Geocortex Viewer for HTML5 allows users to work in environments where connection to the Internet is sporadic or absent. Field workers and remote employees can venture into low-connectivity or no-connectivity areas, and continue to browse and edit spatial and non-spatial data—their normal GIS workflow is not interrupted.


As of version 2.0, to use the Geocortex Mobile App Framework offline, you must use Offline Maps. An Offline Map specifies the feature layers, map area, and offline basemaps that can be made available when the device lacks network connectivity. End users can download an Offline Map created by an administrator to suit their needs, or create their own.

A typical usage scenario would be a field worker who needs to use the Geocortex Mobile App Framework in a remote location lacking network connectivity. Before departing, the worker would download a suitable Offline Map. At the remote location, the worker would activate the Offline Map to continue working with the map, browsing and editing features. Upon returning to a place with network connectivity, the worker can use the Offline Map's **Sync** button  to synchronize any changes to the map with the server.

For a comprehensive list of which features are supported offline, see "Offline Support for Features" in the *Geocortex Mobile App Framework Administrator Guide*.

 From version 2.0 of the Geocortex Viewer for HTML5, offline support requires the Geocortex Mobile App Framework. To download and install the Geocortex Mobile App Framework, visit the [Geocortex Support Center](#).

 Geocortex Workflows are not supported offline.

 In offline mode, buffering operations do not support negative values.

## 18.1 Configure Offline Support

You configure offline support in Manager and the Geocortex Mobile App Framework. For information on offline configuration, see:

- *Geocortex Mobile App Framework Administrator Guide*
- *Geocortex Essentials Administrator Guide*

## 19 Accessibility

**Accessibility** is the degree to which software is accessible to people with disabilities. [Web Content Accessibility Guidelines \(WCAG\) 2.0](#) is a technical standard with the goal of providing a single, shared standard for web content accessibility. WCAG was developed by the Web Accessibility Initiative of the World Wide Web Consortium ([W3C](#)). The Geocortex Viewer for HTML5 2.9 adheres to WCAG 2.0.

There are two aspects to accessibility support in the HTML5 Viewer that end users can use:

- **Screen Readers:** Run a screen reader to vocalize and interpret page content.
- **Keyboard Shortcuts:** Interact with the viewer using only the keyboard.

Screen readers and keyboard shortcuts can be used together.



Users are informed about these accessibility features by the configurable [Accessibility Panel](#).

## 19.1 Screen Readers

Screen readers read user interface (UI) text aloud, so the user can listen to the page instead of seeing it. They do this by monitoring the position of the mouse pointer and reading the text where the pointer is positioned. It does not matter how the pointer is controlled—the user can navigate using a mouse, the keyboard, a sip-and-puff device, or any other type of navigation device.

Screen readers also provide contextual information for the text being read. For example, if the user navigates to the **Sign in** button, a screen reader might say **Sign in button**—"Sign in" is the text that is read, and "button" is the context.

The context is essential. It is how the user knows what will happen when the item is activated—buttons perform an operation, hyperlinks navigate the browser, checkboxes select or clear a setting, and so on.

In order for a user to use a screen reader with an HTML5 Viewer, the user must have a screen reader installed and running when using the viewer. No additional steps are required. The HTML5 Viewer is tested using the Freedom Scientific [JAWS screen reader](#) but others may also work.

In the HTML5 Viewer, there are three pieces of information that are provided about the map:

- The coordinates at the center of the current map extent.
- The current scale of the map. The scale is only provided when the zoom level changes.
- The number of visible features of each visible layer. For example, "There are 20 features visible on World Cities."

To make the screen reader read out information about the map without changing the map extent, select the map with either the mouse or the **TAB** key.

## 19.2 Keyboard Shortcuts

Keyboard shortcuts allow end users to interact with the HTML5 Viewer using a keyboard instead of a mouse. The current UI component is highlighted with a border, which is dark purple by default.

Keyboard shortcuts do not interfere with or prevent the user from using the mouse—the user can go back and forth between keyboard shortcuts and the mouse. The only exception to this is when drawing a shape on the map for an identify, draw, or measure operation—the user cannot switch the type of device part way through drawing the shape.



Once you have selected a tool to draw a shape, to draw the shape with the keyboard, press **Enter**. If you draw the shape with the mouse instead, you will not be able to use the keyboard to create the shape. Note this only applies to when you first create the shape; when you edit a shape, you may use either the keyboard or the mouse.



Operations that use a freehand geometry cannot be done using the keyboard. This includes freehand identify and freehand draw operations.



Keyboard shortcuts provide a level of precision that a mouse does not. You can move or resize a shape by a single pixel using the keyboard.

## General Keyboard Shortcuts

The HTML5 Viewer uses standard shortcuts to navigate the page and select or activate items.

### General Keyboard Shortcuts for Accessibility

To do this...	Press...	With JAWS screen reader...
Navigate forward through the page's components	TAB <sup>1,2</sup>	TAB <sup>1,2</sup>
Navigate backward through the page's components	SHIFT+TAB	SHIFT+TAB
Select or activate the current UI component	ENTER	ENTER
Select checkbox, radio button, or toggle	SPACE BAR	SPACE BAR
Pan the map (if selected)	Arrow keys <sup>3</sup>	ALT+Arrow keys <sup>3</sup>
Move slider	Arrow keys	ALT+Arrow keys
Jump slider	PAGE UP; PAGE DOWN	ALT+PAGE UP; ALT+PAGE DOWN
Jump slider to the start or end	HOME; END	ALT+HOME; ALT+END

<sup>1</sup> The initial tab operation in a new Viewer session opens the skip navigation links menu consisting of links to the **Side Panel**, **Search**, **I want to...**, **Toolbar**, and **Map**. Initial focus is on the **Side Panel** link, and subsequent tabbing traverses each of the items in the menu. Users can quickly return to the skip navigation links menu by clicking the address field in the Viewer and then tabbing to open the menu. Users can also return to the initial item in the skip navigation links menu by continuing to tab through the Viewer, or they can access the last link in the menu by tabbing back (SHIFT+TAB) through the Viewer.

<sup>2</sup> In Chrome, you cannot tab between individual items in a radio group. You must tab to the group, and then use the arrow keys to change the selection.

<sup>3</sup> If the [Accessibility Module](#)'s `expandedMapKeyboardAccessibility` configuration property is `false`, the mouse pointer must hover over the map to pan with the arrow keys. By default, it is `true`.

## HTML5 Viewer Keyboard Shortcuts

The HTML5 Viewer has its own shortcuts for working with shapes, including text markup. These shortcuts are used with tools that require the user to manipulate a shape on the map, specifically, identify, draw, measure and edit tools.





Some of the JAWS screen reader default keyboard shortcuts may override the Viewer's keyboard shortcuts listed in the following table. The user may need to remap the JAWS keyboard shortcuts to ensure that they function in the Viewer as indicated.

### HTML5 Viewer Keyboard Shortcuts for Accessibility

To do this...	Press...	With JAWS screen reader...
Add a vertex to the shape that you are creating	ENTER	ALT+ENTER
Move vertex horizontally or vertically	Arrow keys	ALT+Arrow keys
Move vertex diagonally	PAGE UP; PAGE DOWN; HOME; END	ALT+PAGE UP; ALT+PAGE DOWN; ALT+HOME; ALT+END
Move the selected shape horizontally or vertically	Arrow keys	ALT+Arrow keys
Move the selected shape diagonally	PAGE UP; PAGE DOWN; HOME; END	ALT+PAGE UP; ALT+PAGE DOWN; ALT+HOME; ALT+END
Enlarge the selected shape uniformly	S	ALT+S
Reduce the selected shape uniformly	SHIFT+S	ALT+SHIFT+S
Rotate the selected shape to the right	R	ALT+R
Rotate the selected shape to the left	SHIFT+R	ALT+SHIFT+R
Complete the shape	ENTER, ENTER (press ENTER twice)	ALT+ENTER, ALT+ENTER (press ALT+ENTER twice)
Enter vertex editing mode	V	ALT+V
Select the next vertex of the current shape (in vertex editing mode)	V	ALT+V
Select the previous vertex of the current shape (in vertex editing mode)	SHIFT+V	ALT+SHIFT+V
Delete vertex	D	ALT+D

To do this...	Press...	With JAWS screen reader...
Exit vertex editing mode	ENTER	ALT+ENTER
Enable snapping (when the mouse pointer is over the map while using the Measure, Draw, Edit, Create New Feature, or non-dragging Identify tools)	F	ALT+F



For greater precision when moving, rotating or resizing a shape, hold down the ALT key while pressing the desired shortcut keys. For example, press ALT+LEFT ARROW to move the selected shape one pixel to the left.

### Example 1: Use the Keyboard to Measure Distance

1. Press **TAB** as many times as needed to navigate to the **Measure** multitool.
2. Press **ENTER** to open the **Measure** multitool.  
The pointer is positioned on the Measure Distance tool.
3. Press **ENTER** to activate the **Measure Distance** tool.
4. Press **ENTER** to create the first vertex.  
Because you are drawing a line, the vertex is an endpoint.
5. Use the arrow keys and diagonal movement keys to move the endpoint close to the desired position.
6. Use the **ALT** key in combination with any other movement keys to move the endpoint to the precise position that you want.  
The ALT key restricts the movement to one pixel each key press.
7. Press **ENTER** to mark the position of the first endpoint and create the other endpoint.
8. Move the endpoint to the desired position.
9. Press **ENTER** twice to mark the position of the second endpoint.  
This completes the measurement.

### Example 2: Use the Keyboard to Draw a Polygon

1. Press **TAB** as many times as needed to navigate to the **Draw** multitool.
2. Press **ENTER** to open the **Draw** multitool.
3. Press **TAB** as many times as needed to navigate to the **Polygon** tool.
4. Press **ENTER** to activate the **Polygon** tool.
5. Press **ENTER** to create the first vertex.
6. Use the arrow keys and diagonal movement keys to move the vertex close to the desired position.
7. Use the **ALT** key in combination with any other movement keys to move the vertex to the precise position that you want.

The ALT key restricts the movement to one pixel each key press.

8. Press **ENTER** to mark the position of the first vertex and create the next vertex.
9. Move the vertex to the desired position.
10. Continue adding and positioning vertices until there are no more vertices to add.
11. Press **ENTER** twice to close the polygon.

### Example 3: Use the Keyboard to Move, Rotate, or Resize a Shape

1. Press **TAB** as many times as needed to navigate to the **Edit** tool.
2. Press **ENTER** to activate the **Edit** tool.
3. Press **ENTER** again to draw a point on the map.
4. Move the point to the shape that you want to edit.
5. Press **ENTER** to select the shape that the point is on.
6. Use the move, rotate, and resize keyboard shortcuts to modify the shape.
7. Press **ENTER** twice to finalize the shape's size and position.

#### See Also...

[Accessibility Module on page 131](#)

## 20 Protecting Against Malicious Code

The HTML5 Viewer has a built-in security feature that helps to prevent malicious attacks, in particular cross-site scripting (XSS) attacks. The feature works like this: Before the content loads on the user's device, viewers remove untrustworthy URLs, HTML markup that could execute code when resolved.

In addition, the viewer prompts the user to allow or deny the content at unknown URLs. This protects users from unknowingly loading content that could be malicious. Once a user has allowed or denied content from a particular URL, the viewer remembers the user's choice in the web browser's local storage and does not prompt again for that URL. The user's preferences remain in effect until the user clears their web browser's local storage.

URLs and HTML markup can occur in different contexts. Some common contexts are:

- feature data
- projects
- uploaded data
- site configuration, notably feature descriptions

## What Viewers Remove

HTML5 viewers remove high-risk URLs and HTML markup. The viewer does not notify end users that content is being removed. Viewers remove the following:

- Scripts defined using the HTML `script` element.
- HTML attributes that can run code, for example, JavaScript events like `onload` events.
- HTML attributes that do not conform to the HTML4 specification, for example, `data` attributes.
- Custom tags.
- URLs that use any protocol besides HTTP, HTTPS, or MAILTO. For example, viewers remove URLs that use the FTP protocol.

Most HTML elements are allowed, including `a`, `img`, `iframe`, `video`, and `audio` elements. `a` tags can contain HTML5 viewer commands, even though commands run code. The code that is run by viewer commands is considered to be trusted.

## How Viewers Handle URLs

To help prevent loading malicious code via URLs, HTML5 viewers filter URLs before resolving them. HTML5 viewers filter URLs as follows:

1. Remove URLs that use any protocol besides HTTP, HTTPS, or MAILTO.
2. Create a whitelist of allowed URLs.  
These URLs are trusted. The viewer loads content from trusted URLs without prompting the user.  
for more information, see [URLs that are Explicitly Specified in the Site on page 409](#).
3. Before loading content from a URL that is not trusted, prompt the user to allow or deny content from that URL.  
You can disable the prompt from the viewer configuration files. See [Content Policy Configuration on page 409](#).
4. Remember the user's choice.  
If the user allowed the content, the viewer will load content from that URL without re-prompting the user.  
If the user denied the content, the viewer will not load content from that URL. The viewer does not re-prompt the user.



The viewer remembers the user's preference relative to the URL's specificity. If the URL for a site's subdirectory is trusted, then every file within the subdirectory is also trusted. Additionally, if a URL that specifies a specific filename within the subdirectory is not trusted, only that file is not trusted.

## Allow Unsafe Content

The Application page in a viewer's Management Pack has a checkbox labelled Allow Unsafe Content. The viewer prevents XSS attacks and filters URLs and markup whether or not this setting is on.

If this checkbox is selected, content from a KML or GeoRSS layer that contains HTML markup within a `<description>` is interpreted by the viewer. If you want the viewer to display the preformatted markup, clear the checkbox. By default, unsafe content is not allowed.

## Content Policy Configuration

You can enable or disable the prompt users receive when asked to allow or deny content from a URL. To configure the security prompt, locate the `application` object near the top of your viewer configuration files and embed a new object called `contentPolicy` in it. The `contentPolicy` object has these configurable properties:

- `disableSecurityPrompt`: Disable the security prompt site-wide. The default value is `false`.
- `disablePromptOnImages`: Disable the security prompt on images. The default value is `true`.
- `trustExactUrl`: For security prompts, only trust the exact URL instead of any URL in the same directory. The default value is `false`.

The configured `contentPolicy` object looks like this:

```
"application": {
  "contentPolicy": {
    "disableSecurityPrompt": false,
    "disablePromptOnImages": true,
    "trustExactUrl": false
  }
}
```

For more information about viewer configuration files, see [About Viewer Configuration on page 57](#).

## URLs that are Explicitly Specified in the Site

HTML5 viewers always allow content from URLs that are configured explicitly in the site. This includes the URLs to map services, geocoding and geometry services, feature hyperlinks, layer hyperlinks, and icon URLs. As well, viewers allow any URLs (or parts of URLs) that are explicitly configured in a layer's Feature Label, Feature Description, or Feature Long Description. This is the viewer's way of respecting the administrator's configuration.

The key here is that the configuration must be explicit. Replacement tokens are not considered to be explicit. Viewers allow content up to the last slash before the first replacement token, or if there is no replacement token, up to the last slash. For example:

- `https://server.domain.com/content/`  
The viewer allows all files in `https://server.domain.com/content/` without prompting the user.
- `https://server.domain.com/content/file.extension`  
The viewer allows all files in `https://server.domain.com/content/` without prompting the user.
- `https://server.domain.com/content/{name}.extension`  
The viewer allows all files in `https://server.domain.com/content/` without prompting the user. `{name}` is a field token.
- `https://server.domain.com/ArcGIS/rest/services/mymapservice/{LayerID}/{OBJECTID}/attachments/{PHOTOID}`  
The viewer allows all files in `https://server.domain.com/ArcGIS/rest/services/mymapservice/` without prompting the user. `{LayerID}` is a layer token.

## 21 Translation

### 21.1 About Translating UI Text

The HTML5 Viewer is designed to facilitate translation of the viewer's user interface text. To support translation, the user interface text for each library is located in two language files.

To translate the viewer, you make a copy of the language files you want to translate, name them with the appropriate language tag, and then translate the text in the files. When the viewer runs in a browser, the viewer detects the browser's language setting and looks up the text in the language file with the matching language tag. Each text item in a language file is assigned to a placeholder called a "key". The code for the HTML5 Viewer references the keys, not the text. When the viewer runs in a browser, the viewer looks up the key in the appropriate language file and displays the text assigned to that key.

Normally, you do not need to change the keys that are configured—you change the text, not the key. You use the default keys for all languages. For example, the key for the title of the I Want To menu, `@language-iwtm-title`, is assigned the value "I Want To..." in the default (English) language file. If your viewer is also available in French, `@language-iwtm-title` might be assigned "Je veux..." in the French language file.

The default language for the HTML5 viewer is US English (language tag "en-US"). The first part of a language tag is a language code ("en") from the [ISO 639-1 standard](#). The second part of a language tag is a country code ("US") from the [ISO 3166-1 standard](#).

### 21.2 Translate Language Files

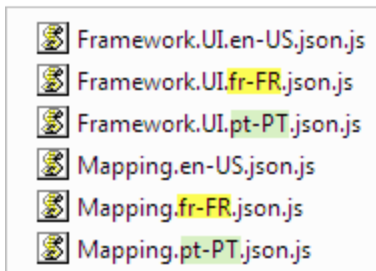
There are two language files that contain the strings that appear on the application interface. The files are:

- `Framework.UI.en-US.json.js`
- `Mapping.en-US.json.js`

#### To translate the language files:

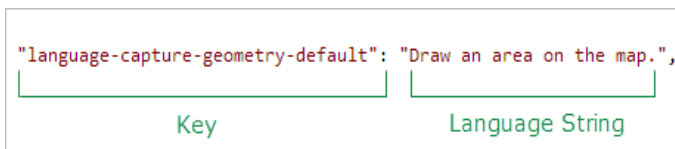
1. Navigate to the `Locales` folder containing the language files.  
The language files are in the `\Resources\Locales` subfolder of the web folder where you deployed the viewer. For example, if you deployed the viewer to `C:\inetpub\wwwroot\myviewer`, then the files are in `C:\inetpub\wwwroot\myviewer\Resources\Locales`.
2. Create a copy of each language file.

- Rename the language files with the language code of the language you are translating the viewer to.



#### Language files copied and renamed for French and Portuguese

- Open each language file in a text editor and translate each language string.



Be careful to translate the language *strings* and not the *keys*. If you delete a key or even a comma outside of the language strings, this can cause errors and the viewer may not function.



The language files use UTF-8 encoding. If you save the files using the editor's Save As function, make sure you select UTF-8 as the encoding—under other encodings, the viewer's UI strings may contain unexpected characters.

- Open the three configuration files, `Desktop.json.js`, `Handheld.json.js`, and `Tablet.json.js`, in a text editor such as Notepad.

The configuration files are in the viewer's virtual directory. The default location is:

```
C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials\
Default\RESElements\Sites\

```

For more information, see [File Locations on page 448](#).

- In each of the configuration files, each library's `locales` array needs an array item for each language you are going to support. Copy the existing array items as many times as you need to.

The screen capture below shows each library's array item for the default language. The parts that you will change are also indicated.

```
"libraries": [  
  {  
    "id": "Framework.UI",  
    "uri": "Resources/Compiled/Framework.UI.js",  
    "locales": [  
      {  
        "locale": "en-US",  
        "uri": "Resources/Locales/Framework.UI.en-US.json.js"  
      }  
    ]  
  },  
  {  
    "id": "Mapping",  
    "uri": "Resources/Compiled/Mapping.js",  
    "locales": [  
      {  
        "locale": "en-US",  
        "uri": "Resources/Locales/Mapping.en-US.json.js"  
      }  
    ]  
  }  
],
```

**Default language specified in the configuration of the libraries**

7. Add a comma between each pair of array items, and make sure that there is *no* comma after the last array item.



- Change the language codes in the `locale` and `uri` properties so that each library has an array item for each supported language.

```

"libraries": [
  {
    "id": "Framework.UI",
    "uri": "Resources/Compiled/Framework.UI.js",
    "locales":
    [
      {
        "locale": "en-US",
        "uri": "Resources/Locales/Framework.UI.en-US.json.js"
      },
      {
        "locale": "fr-FR",
        "uri": "Resources/Locales/Framework.UI.fr-FR.json.js"
      },
      {
        "locale": "pt-PT",
        "uri": "Resources/Locales/Framework.UI.pt-PT.json.js"
      }
    ]
  },
  {
    "id": "Mapping",
    "uri": "Resources/Compiled/Mapping.js",
    "locales":
    [
      {
        "locale": "en-US",
        "uri": "Resources/Locales/Mapping.en-US.json.js"
      },
      {
        "locale": "fr-FR",
        "uri": "Resources/Locales/Mapping.fr-FR.json.js"
      },
      {
        "locale": "pt-PT",
        "uri": "Resources/Locales/Mapping.pt-PT.json.js"
      }
    ]
  }
],

```

Three languages specified in the configuration of the libraries

- Save the configuration files and test the viewers.

## 22 Custom Development

### 22.1 About Custom Development

Prior versions of the Geocortex Viewer for HTML5 referenced the online Esri ArcGIS API for JavaScript. As of version 2.5, the HTML5 Viewer also includes a partial copy of the ArcGIS API for JavaScript. When the `geocortexUseLocalEsriApi` variable in `Index.html`, `Tablet.html` or `Handheld.html` is set to `true`, the viewer switches from using the online API to the local copy. These files are located in the root folder of where the HTML5 Viewer is installed. The Geocortex Mobile App Framework requires the local copy of the ArcGIS API to function. By default, viewers downloaded by the Geocortex Mobile App Framework have this variable set to `true`.

If you engage in custom development, it is important to copy any necessary ArcGIS API files that you use to your local copy, in one of the following folders within the HTML5 Viewer folder as appropriate:

- Resources/Scripts/dijit
- Resources/Scripts/dojo
- Resources/Scripts/esri

If you do not, your viewer application may work in a web browser when referencing the online ArcGIS API, but fail when referencing the local copy, as the Geocortex Mobile App Framework does.



We recommend you set the `geocortexUseLocalEsriApi` variable to `true` when testing your custom development so that missing files will be reported in your web browser console. This will help to ensure your application will work in the Geocortex Mobile App Framework.



Within `Index.html`, `Tablet.html` and `Handheld.html`, you can simply uncomment the line that reads `//var geocortexUseLocalEsriApi = true;` by removing the two preceding forward slashes to set the variable.

For example, after setting the `geocortexUseLocalEsriApi` variable to `true`, suppose your web browser console indicates `dijit/tree/TreeStoreModel.js` is missing. Download the file from <http://js.arcgis.com/3.20/dijit/tree/TreeStoreModel.js> and save it to `Resources/Scripts/dijit/tree/TreeStoreModel.js` within your HTML5 Viewer folder.

## 22.2 Key Concepts

### 22.2.1 HTML and JavaScript

HTML and JavaScript applications have traditionally been difficult to organize and maintain because they do not come with any built-in organizational tools or guidelines. As a result, web applications have a tendency become large collections of interdependent files with markup and business logic spread throughout.

However, if you apply proven organizational and architectural concepts, and you enforce intelligent conventions, you can greatly improve the quality and life cycle of JavaScript applications and produce solutions that are as clean, robust, and maintainable as in any other language or platform.

### 22.2.2 Separation of Concerns

A clean separation and delineation of business logic and presentation code is desirable in almost any software application. Client-side web applications have historically been quite bad at this, and it is a significant source of pain for those tasked with debugging, maintaining, and augmenting existing JavaScript applications.

Web applications commonly mix markup and script code across both HTML and JavaScript files. It is quite common to see large portions of user interfaces generated and styled in code that is buried in a script file for a future developer to hunt down. Presentation code is quite often mixed and interwoven with business logic.

When you separate user-interface markup, styling, and code from its underlying business logic, web applications become easier to understand and maintain.

In order to facilitate this separation of concerns, the Geocortex HTML5 framework separates resources into the following groups:

- View markup (HTML)
- View styles (CSS)
- View code (JavaScript)
- Business Code (JavaScript)

These application resources interact with each other so that they function as a cohesive unit, while being separate both logically and semantically in the implementation.

There are two key concepts that allow this separation to work effectively: [MVVM \(Model-View-ViewModel\)](#) and [Data Binding](#).

### 22.2.3 Model-View-ViewModel (MVVM)

Model-View-ViewModel is an architectural pattern based on Model-View-Controller (MVC), a pattern for developing user interfaces that separates the internal representations of information from the way information is presented to or accepted from the user.



There are many good online resources to familiarize you with MVVM. We recommend becoming familiar with the basic concepts as a minimum. The HTML5 Viewer SDK includes samples of MVVM in action.

There are three distinct entities in the MVVM pattern:

- The **View**: A visual component that presents information from a View Model.
- The **View Model**: An abstraction of data and state that is easily consumable by the View.
- The **Model**: The underlying data or entity modeled by View Models.

### Example Comparison of Traditional and MVVM Applications

An example might be an interface component that displays a list of parcels that meet a set of criteria.

In a traditional web application, a parcel list user interface would be generated programmatically by iterating over a number of parcel objects and manually building HTML markup using string concatenation and/or Document Object Model (DOM) manipulation. This code would be in the web page, in script tags, or it may be buried deep in a script file. The styles for this component would be to be in a master Cascading Style Sheet (CSS) file, or even declared inline where the user interface markup is generated.

In a Geocortex HTML5 application, the parcel list component is separated into its constituent parts. The folder containing the source for this Module would contain the following:

- Parcels.css
- ParcelsView.html
- ParcelsView.js
- ParcelsViewModel.js

Even for those not familiar with MVVM, the duties of each file should be somewhat clear. CSS styles and HTML markup are separated into their own files and define the layout and style of the interface component. The User Interface (UI) code and business code is not mixed. The UI code for the parcel list interface component has its own code file (ParcelsView.js). The business code (ParcelsViewModel.js) also has its own file, as the two have separate concerns.

### Interaction

The Geocortex HTML5 framework provides mechanisms that allow these pieces to work together while being defined separately. A data-binding engine allows View markup to bind to fields in the View Model and event handlers in the View. In addition, View code can access View Model members and can call View Model methods. View markup is styled by View CSS.

This organizational style of UI development lends itself well to rapid development and the creation and maintenance of reusable controls and widgets.

## 22.2.4 Data Binding in HTML

The other key component that allows for a separation of concerns is data binding. Data binding provides a clean and powerful way to create user interfaces without writing boilerplate code that unnecessarily queries and modifies the DOM.

Data binding provides the glue between Views and View Models. By using data binding expressions, developers are able to declaratively specify relationships between HTML elements, View Model properties, and View events.

### Example Comparison Continued

We can continue with the example of the component that displays a list of parcels.

In traditional client-side web development, the parcel list user interface is typically created and updated manually to reflect the underlying list of parcels. Libraries like jQuery and Dojo make this kind of task easy.

Consider a typical line of jQuery, used to append a parcel to our list:

```
$(".parcels ul").append("<li>" + parcel.ownername + "</li>");
```

This code looks fine - it is simple and terse - but it suffers from significant drawbacks. For example, every time this code is run to add a new parcel to the list, it is probably invoking the selector for `.parcels ul`. Depending on the complexity of the page, this can be a lot of overhead for a task that may be repeated hundreds or thousands of times, especially if we are searching from the root of the page. On a mobile device, this becomes a significant factor.

In addition, this code example mixes markup and code. What happens if a designer or developer wants to change from using a list to using a table? They would have to hunt down this piece of code, whose location probably has little correlation with what is seen in a DOM inspector tool.



An experienced JavaScript developer might point out that in this example the selector could be pre-cached, the query limited to a particular element, and that a call to `end()` would save unnecessary work. The data binding engine performs this type of work under the hood so that developers don't have to.

A much better solution to the traditional style of DOM manipulation would be to remove the burden of it from the developer and have the system perform it in an efficient manner.

In an MVVM implementation the parcels View markup would be:

```
<div class="parcels">
  <ul data-binding="{@source: parcels}">
    <!-- Template item for each member of the parcels collection. -->
    <li data-binding="{@template-for: parcels}{@text: ownerName}"></li>
  </ul>
</div>
```

In the Geocortex HTML5 framework, this code example has the same functionality as the previous hypothetical piece of sample traditional code. The difference is that there is no developer code involved in populating the UI. The relationship between data and the presentation is declared in the View markup.

The Geocortex HTML5 framework introduces a custom `data-binding` HTML attribute that declaratively associates UI elements with View Model fields and event handlers in the View code.

In the above example, the unordered list `<ul>` has a `@source` binding to a `parcels` collection of an underlying View Model, and contains a single list item `<li>` as a template that has a `@text` binding to an `ownerName` attribute.

For each member of the parcels collection, a list item `<li>` element is created and its own binding expressions satisfied against each parcel object in the collection.

The end result of this example is a list of parcel owner names. This UI component automatically updates whenever the underlying collection of parcels is changed in some way.



Data binding expressions are covered in depth later in this guide.

The data binding engine picks up binding expressions when UI markup is added to the page as a View. Views are inspected and all binding expressions parsed and resolved up front, leaving the developer free to deal with business logic instead of writing UI event handlers and worrying about other tedious interface considerations.

The data binding approach is effective to the developer primarily because he or she escapes the time costs in writing and maintaining brittle UI manipulation code. It is also beneficial to mobile performance, as bindings are wired up once by the binding engine and resolved into event handlers and closures behind the scenes. The structure of View HTML does not need to be repeatedly queried, and modification is done effectively by the system, resulting in a UI that is efficient at both design time and run time.

## 22.2.5 Libraries and Resource Compilation

A well-organized web application will typically have many files of varying types. An application may have hundreds or thousands of code files, CSS style sheets, and HTML views.


The Geocortex SDK for HTML5 contains a tool called the Resource Compiler Tool. This Java-based command line tool examines simple XML build manifest files and combines the listed resources listed into a library. A library is a collection of code, CSS, and HTML files that are combined into a single JavaScript file.


By amalgamating most or all of an application's resources into one file, you can avoid invoking many HTTP requests to fetch application resources. This behavior is particularly desirable when building mobile and offline applications.

The Resource Compiler Tool is generally run as a build step to compile and output library files in a desired location.

The Resource Compiler Tool can optionally use Google's Closure Compiler to perform minification, JavaScript optimization, and warning of potential bugs and various caveats.

The QuickStart that is included in the SDK package contains a sample build script and demonstrates use of the Resource Compiler Tool.

 Java 5 or greater is required to run the Resource Compiler Tool. Java must be on your PATH environment variable in order to run the SDK build scripts.

 For information on how to use the Resource Compiler Tool, run the tool without any parameters. For example, `java -jar ResourceCompilerTool.jar`.

See also...

[Resource Compiler Tool on page 425](#)

## 22.2.6 UI Composition

### Views and Regions

A view is a user interface component. It can be anything from a sophisticated AJAX form to a small piece of static HTML content. Views are typically hosted inside of named regions, but may live outside of a region. Views typically use data-binding expressions in their markup to represent data and state of associated view models. This helps facilitate the separation of concerns.

A region is an area of a visual element that has a name associated with it. Regions are used to define the style and behavior of a particular area of a UI interface. An example would be an area in which a map is typically displayed, or a collapsible area where search results or forms are displayed. For a list of the regions in the different shells, see [Regions on page 444](#).

The way a particular region behaves is dictated by its region adapter. A region adapter is a piece of code that displays one or more views in a specific fashion for a given region. An example of a region adapter is an adapter that shows multiple views in an HTML `div` region and allows a user to switch between them using tabs.

Regions are declared in view markup using the “`data-region-name`” attribute on any HTML element that can hold children, for example,

```
<div class="application-Region" data-region-name="ApplicationRegion"></div>
```

By default, regions use an adapter that simply shows one view at a time in the given region.

Views typically hold references to the region that they are hosted in. When a view is *activated*, it is generally added to the desired region and made visible, although the exact behavior depends on the type of region adapter used.

If a view references a region that does not exist, the view is not displayed until that region is created. This is a useful mechanism that allows flexible UI configuration. Views in configuration can be declared before regions that host them are declared, and they are then only created when those regions are available.

Views themselves can contain regions and views may move between regions.



Views can be associated with regions via configuration, or created and placed on the fly programmatically using the View Manager object (`viewManager`) of an application. When creating views (and view models) on the fly, be sure to pass along the Application instance, for example, `this.app`; and the Library ID, for example, `this.LibraryId` of the component doing the creating.

When a region is destroyed, any views inside it are also destroyed.

Views are intended to be flexible and used in the place of manual DOM creation. Views can be created without being associated with regions or view models.



Instead of using string concatenation and/or DOM manipulation to create UI elements, views and data-binding should be used. Using the view-oriented method of UI creation has serious benefits over programmatically creating DOM elements. Views should be the source of nearly all UI markup.

### Liquid Layouts

We strongly suggest that developers use liquid layouts when creating views. Liquid layouts are layouts that expand to take up as much space as they are given, and generally have no explicit width or height constraints.



When creating user interfaces for cross-platform applications, avoid checking user-agent strings and creating phone/tablet/desktop specific code conditions. Instead, organize your views and regions in ways that allow them to be configured to meet layout requirements. For example, if you have a view that displays complex results, consider breaking it into two views: a table view for desktops and tablets, and a list view for phones. A developer or administrator can then choose which one to use via configuration, instead of depending on code to detect the appropriate setup.

By creating liquid views, developers can create flexible and reusable user interfaces that work across multiple device form factors.

An example of this is a simple menu view. On a large-format device such as a desktop PC, we may wish to have a menu pop up over the map and take up a small area of the screen to display some options. However, on a smaller device like a phone, we may wish to have the menu take up the whole width and height of the screen in order to be touch-friendly. The width and height of the region holding this menu view may differ between devices, but the view itself should simply take up whatever space is available to it.



When creating user interfaces for use on a multitude of device sizes, ensure that it is the layout and styling of the regions that change, rather than the views. Platform-specific concerns should live in configuration files and CSS style sheets, not in code.

#### 22.2.6.1 Shells

A shell can be thought of as the basic layout of an application. It is the skeleton of a user interface. A shell is typically just a view in which various application regions are defined along with which region adapters are used. A shell view may also have custom code to handle certain behaviors, such as transitions and animations between certain states or layouts.

The HTML5 Viewer ships with three shells:

- Desktop shell geared towards devices with large screens.
- Tablet shell for large and medium touch-screen devices.
- Handheld shell geared towards smaller touch-screen devices such as smartphones.

The shell for both Desktop and Tablet contain the same regions. The shell for Handheld devices contains only essential regions. For a list of the regions in the different shells, see [Regions on page 444](#).

## 22.2.7 Commands and Events

**Commands** provide a mechanism that allows application components to invoke functionality without knowing who the actual implementer(s) may be.

Commands are invoked by name and can have zero or more implementations registered. These implementations will be executed sequentially when a command is invoked.

Commands do not return results, and should generally not modify their arguments.

Since commands can be invoked from workflows via the `RunExternalCommand` activity, it is better to keep command parameters as simple as possible.

Commands live inside of the application's `CommandRegistry` object. Referencing a non-existent command simply creates a new empty command.

**Events** are similar to commands. Events allow components to respond to events in a system. Events, like commands, are referenced by name and contain zero or more pieces of functionality that are executed sequentially when the event is published.

Events differ from regular JavaScript and Dojo events in that they are not global. Geocortex Events live inside the scope of an Application instance and so multiple applications can live on a page without interfering with each other's events.

See the SDK Sample **Commands and Events** for examples of how to create and invoke commands and events.



Modules that interact with each other should always do so via commands and events. Modules should never have explicit coupling. Removing one module should not break another.

See also...

[Geocortex SDK for HTML5 API Reference on page 422](#)

## 22.2.8 States

Geocortex Viewer for HTML5 2.5 introduces the notion of application states. States are a means to provide context as to what is currently happening in the viewer. States are entered or exited when a certain command runs, a specific event occurs, or in response to user activity. For example, `IdentifyState` is activated when the Identify Tool becomes active.

There are two kinds of states:

- **Global States:** Only a single global state can be active at a time. If a new global state activates, the previous one becomes inactive. For example, `IdentifyState` cannot be active at the same time as `MeasureState`. A global



state can, however, be active at the same time as any number of non-global states.

- **Non-global States:** Any number of non-global states can be active simultaneously - regardless of whether a global state is active or not. For example, `TransientActiveState` and `SnappingState` can be active simultaneously.

There is also a default non-modal state, `DefaultState`, which is always active, even when no other states are active.

States are typically associated with:

- [Toggle buttons](#), which can be configured in Essentials Manager.
- Context-sensitive toolbars, which are configured in the `transientElements` array of both the [Tabbed Toolbar](#) and the [Compact Toolbar](#). These toolbars may contain buttons or toggle buttons.

For a complete list of application states, see the [State Reference on page 452](#).

See also...

[Configure the Toolbar on page 108](#)

[CompactToolbar Module on page 150](#)

[TabbedToolbar Module on page 363](#)

[State Reference on page 452](#)

## 22.3 Samples Viewer

The Geocortex Viewer for HTML5 installation package includes a Samples Viewer, which contains both the Geocortex SDK for HTML5 API Reference and interactive samples with source code. The Samples Viewer is intended to be hosted on a web server such as Internet Information Services (IIS). Alternatively, you can [view the Samples Viewer for the latest Geocortex SDK for HTML5 version online](#).

### ► To host the Samples Viewer on a web server:

1. Download and extract the [Geocortex Viewer for HTML5 Installation Package](#) from the Geocortex Support Center.
2. Navigate to the folder where you extracted the contents of the Installation Package ZIP file.
3. Open the **Samples** folder.
4. Copy the **SamplesViewer** folder to a folder hosted by your web server. In Internet Information Services (IIS), the default location for hosted content is:

```
C:\inetpub\wwwroot
```

You can now access the Samples Viewer by opening one of the following URLs in your web browser:

- `http://MyServer.com/SamplesViewer`

This URL will work on all computers that have access to the web server. Substitute `MyServer.com` with the domain of your web server.

- `http://localhost/SamplesViewer`

This URL will only work on the same computer as the web server.



The sample named **Using Modules in Web AppBuilder for ArcGIS** will only load if the domain in the URL is associated with a valid SSL certificate.

## 22.4 Geocortex SDK for HTML5 API Reference

The Geocortex SDK for HTML5 API Reference provides detailed information about the Geocortex APIs (Application Programming Interfaces), including information about namespaces, objects, methods, properties, parameters, commands and events. The Geocortex SDK for HTML5 API Reference covers the following aspects of the Geocortex SDK (Software Development Kit):

- **Geocortex Viewer for HTML5:** A highly configurable and extensible web-mapping application that is built on top of the following APIs.
- **Geocortex Essentials JavaScript API:** A client-side companion to the Geocortex Essentials REST API, allowing you to build mobile and desktop web-mapping solutions by interacting with Geocortex Essentials and the ArcGIS Platform.
- **Geocortex HTML5 Framework API:** A modern web development framework that includes support for MVVM (Model-View-ViewModel) and data-binding.

There are a few ways to access the Geocortex SDK for HTML5 API Reference:

- [Access the latest version of the Samples Viewer online](#)
- [Host and access the Samples Viewer on a web server](#)
- [Access a local copy of the Geocortex SDK for HTML5 API Reference](#)

### ▶ To access the latest version of the Samples Viewer online:

1. Open the [Samples Viewer for the latest Geocortex SDK for HTML5 version online](#).
2. Click the **Documentation** tab.
3. Navigate to the section of interest via the links on the page: for example, **commands** or **events**.

### ▶ To host and access the Samples Viewer on a web server:

1. Follow the [instructions to host the Samples Viewer](#) on a web server.
2. Click the **Documentation** tab.
3. Navigate to the section of interest via the links on the page: for example, **commands** or **events**.

### ▶ To access a local copy of the Geocortex SDK for HTML5 API Reference:

1. Download and extract the [Geocortex Viewer for HTML5 Installation Package](#) from the Geocortex Support Center.
2. Navigate to the folder where you extracted the contents of the Installation Package ZIP file.
3. Open the **Geocortex SDK for HTML5 API Reference** shortcut.  
The Geocortex SDK for HTML5 API Reference will open in your default browser.



In Internet Explorer, you may be asked to allow the running of scripts. If so, click **Allow blocked content**.

4. Navigate to the section of interest via the links on the page: for example, **commands** or **events**.



While the Geocortex SDK for HTML5 API Reference can be viewed offline in the manner above, it was designed to be viewed as a hosted website.

## 22.5 SDK and QuickStart

The Geocortex HTML5 SDK (Software Development Kit) package has everything you need to begin developing web mapping applications using the Geocortex HTML5 framework.

The SDK package contains a number of development samples as well as a QuickStart package. The QuickStart is a ready-to-go application, designed to get you up and running as fast as possible with custom development. The QuickStart also serves as an example of a good way to lay out a web application project using the HTML framework.

This section assumes that you are using the QuickStart package and describes the process of using the QuickStart to create a custom Geocortex HTML5 application.

### 22.5.1 Basic Viewer

If you want to set up a basic HTML5 viewer and point it to a site, the SDK package contains a viewer zip file that contains a basic viewer. Extract the viewer, open the Desktop, Tablet, or Handheld configuration files located under `Resources/Config/Defaults` in the viewer's web folder, and change the `siteUri` parameter of the Site Module to point at the desired site.

Another alternative is to use the VTE file included in the SDK package to create and configure viewers using Manager.

**See also...**

[Geocortex SDK for HTML5 API Reference on page 422](#)

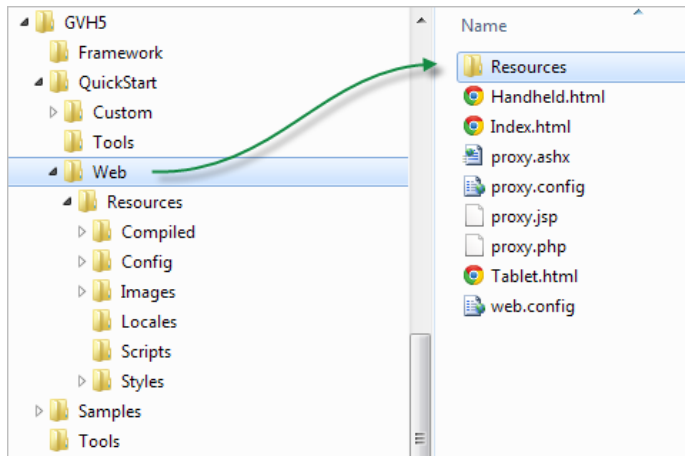
## 22.6 Project Layout

Geocortex HTML5 applications follow a general project layout that aims to keep development clean and organized.

Projects typically have a `Web` or `Viewer` folder. This folder can be considered the output folder. This is the folder that will be hosted by your web server.

A `Web` folder typically holds a standard Viewer deployment. Compiled Library code and resources are copied into the `Web` folder under a `Libraries` subfolder the first time the build script runs. The `Libraries` folder usually holds a number of sub-folders itself, each one corresponding to a custom Library.

The `Web` folder also holds a `Resources` folder. The `Resources` folder is for application-wide resources that are used globally by multiple Libraries. If you open the `Resources` folder, you'll notice a number of subfolders, each holding a different type of resource. Library folders follow this same convention. The `Resources` folder holds compiled code, configuration, images, and styles to be used by the application itself.



### The Resources folder under the Web contains different types of resource

At the root of the QuickStart, there are two more folders. The `Custom` folder is a source folder for the Custom Library, pre-made for the purpose of the QuickStart, and it holds template code to get developers started with Module development.

The `Custom` folder is structured as a typical `Library` folder. It contains an XML resource manifest that the Resource Compiler uses to generate the compiled Library, a `Modules` folder that holds Module-specific code and resources, and a `Resources` folder that holds Library-wide resources.

When the build script executes, the contents of the Custom Library folder is assembled and dropped into the `Libraries` folder of the `Web` folder.

The QuickStart also contains a `Tools` folder that holds the Resource Compiler tool, and the Google Closure Compiler used for optimization and minification of JavaScript files.

## 22.7 Build

The QuickStart includes two build scripts: a Batch file (`build.bat`) for Windows users, and a shell script (`build.sh`) for Unix/Linux users. Each build script invokes the Resource Compiler against the manifest (`ResourceManifest.xml`) found in the Custom Library, and then copies output files to the `Web` folder.

Once a build script is run, the `Web` folder will contain a `Libraries` folder. The `Libraries` folder will contain the output of the Custom Library.

If you host the `Web` folder and start the viewer, you should see a View from the Custom Library.

Depending on your IDE, you may wish to create a new solution/project first and then copy the QuickStart to that location.

## 22.8 Application Life Cycle

Each Geocortex HTML5 application is based around an Application object that consumes a configuration file and then presents the application as it was configured by an administrator or developer.

Unlike typical JavaScript web applications, Geocortex HTML5 Applications are designed to take up entire pages, be embedded in existing pages, or live side-by-side with other Geocortex HTML5 applications. Applications can be hosted entirely inside of a single HTML element, and are designed to offer interoperability and the ability to embed into any existing web application.

Application instances are created by either passing a configuration URI string or configuration object into the constructor. A call to the application's `initialize` method will begin the loading process. If a URI was passed into the constructor, the URI will be fetched and the configuration JSON parsed. If a configuration object was passed, it will be consumed.

The application will consume configuration materials and begin downloading any Libraries specified within the configuration.

When a Library has been downloaded, any modules that are associated with the Library are loaded, along with any Views and View Models configured for that module. The next Library will then be downloaded and the process repeated.

After an application has completed initializing itself, it fires an initialization callback.



An application may be interactive before being considered fully initialized.

When you are finished with an application, or wish to otherwise terminate it, call the `shutdown` method of the application. Shutting down an application destroys all Regions and Views, disposes of bindings and event handlers, and generally leaves the page in the same state as before it was run.

## 22.9 Some Notes on JavaScript

JavaScript is one of the most commonly misunderstood languages in programming history. It also happens to be the ubiquitous language of the web.

The Geocortex HTML5 framework uses JavaScript in a manner that aims to be as simple and effective as possible.

The Geocortex HTML5 framework uses the Dojo Library for a few key concepts.

The Geocortex HTML5 Viewer itself uses jQuery for some light UI work.

JavaScript is a prototypical object-oriented language. It does not have conventional classes, yet it is a fully object-oriented language. This tends to mislead developers who don't fully understand prototypical inheritance and object composition.

One of the places that Dojo is most frequently used is when declaring object types using the `dojo.declare` method. This method creates objects that behave similar to traditional classes seen in other languages.

For more information about Dojo, see <http://dojotoolkit.org/>.

For more information about jQuery, see <http://jquery.com/>.

## 22.10 The Resource Compiler

### 22.10.1 Resource Compiler Tool

The resource compiler tool included with the Geocortex HTML5 SDK is designed to provide developers with a simple command-line tool to perform the amalgamation of code, markup, and style resources into Library files.

The resource compiler tool integrates with Google's Closure Compiler tool, offering JavaScript analysis, optimization, and minification.

The QuickStart includes an example of resource compiler usage in the form of a build script (`build.bat` for Windows or `build.sh` for Unix/Linux).

## 22.10.2 Resource Manifests

The resource compiler loads XML manifest files that dictate which files are to be included or excluded from Library compilation. Each resource manifest contains a number of Profiles, each one identified by a key.

Here is an example of a typical manifest:

```
<Profile Key="Mapping" Library="true">
  <Code>
    <Include Path="Mapping\Modules\**\*.js" />
  </Code>
  <Markup>
    <Include Path="Mapping\Modules\**\*.html" />
  </Markup>
  <Styles>
    <Include Path="Mapping\Modules\**\*.css" />
  </Styles>
</Profile>
```

This resource manifest defines a Library called `Mapping`. It is marked as an application Library. There is also a `Remove` directive (with an equivalent `Path` attribute) that allows the exclusion of files. Both directives accept wild cards. Files are only included in a Library once, even if picked up multiple times by a rule.



The `*` wild card matches zero or more characters. The `**` wild card matches a partial path; in other words, it matches any folder or subfolder(s) within.

## 22.11 Data Binding

### 22.11.1 About Data Binding

The Geocortex HTML5 framework introduces a powerful data-binding engine, allowing users to focus on writing purpose-specific business code while avoiding tedious, hard-to-maintain DOM code.

This section will cover basic data-binding expressions. For more examples, see the SDK Samples.

### 22.11.2 Basics

Data binding typically occurs when a View is added to a region, and a View Model is associated with that View.

Binding is performed by the “attach” method of the View base class. Binding expressions are used in `data-binding` HTML attributes in View markup. Binding expressions are resolved during the attach phase, and a tree of binding expressions is created and stored in memory.

Binding expressions take this form:

```
{DIRECTIVE: SOURCE}
```

Multiple binding expressions can be present in a single data-binding attribute, and white space between binding expressions is ignored.

The **directive** portion of a binding expression dictates the type of binding to create:

- A directive with an `@` symbol in front of it is known as a “pseudo” binding. Pseudo bindings have different behaviors and typically involve special logic to satisfy the binding. For example, the `@text` directive binds the text of the element to a source.
- A directive without an `@` symbol represents an attribute binding with the directive body representing a DOM attribute by name. See [Attribute Binding on page 428](#). For example, the `href` directive binds the element's hypertext reference (URL) to a source.

The **source** of a binding represents the piece of information (or abstraction) that is reflected by the user interface element. The binding source is typically a View Model field, or the name of an event handler in the View code, depending on the type of binding.

## 22.11.3 The Observable Type

The Geocortex HTML5 framework uses the concept of **observable** properties to facilitate data binding and to provide clean declarative relationships between View markup, View logic, and View Models.

Regular JavaScript variables, while powerful given the language's functional and object-oriented abilities, provide no universal mechanism to detect changes to variables.

In a data bound system, changes to View Model state must automatically update the user interface. However, without a proper mechanism to notify consumers of changes, this becomes tricky.

To provide this behavior, two special types of framework object are used: `Observable` and `ObservableCollection`.

`Observable` represents a single field value. When the value of this field is to be updated, a `set` method is called on the `Observable`. Likewise, when the value of this field is to be fetched, a `get` method is called.

`Observable` is a wrapper around instances of `Object`.

`Observable` objects possess an internal `Event` object and event handlers can be bound to the `Observable` to notify interested parties of changes using a `bind` method. The `bind` method subscribes an event handler and executes it in a given scope when the `set` method is called.

See the SDK reference for more `Observable` details.

When an `Observable` is modified using `set()`, any subscribers are automatically notified of the new value.

This concept lies at the heart of data binding and enables creation of sophisticated user interfaces without the need to write DOM manipulation code.

The pattern for creating `Observable` objects is as follows:

```
dojo.declare("myNamespace.myViewModel", null, {
  title: null,
  constructor: function () {
    this.title = new Observable();
  },
  initialize: function (config) {
    this.title.set(config.title);
  }
});
```

`Observable` objects should always be initially set to `null`, and should always be initialized in the `constructor` of an object.

Developers should take care to always use `set()` on `Observable` objects, as regular assignment will not work and will break bindings.

While binding to regular, non-`Observable` JavaScript variables will work in some cases, the binding will be a one-time binding and updates to the source variables will not be reflected in the UI.

All View Model fields intended to be public should be instances of `Observable` or `ObservableCollection`.

#### 22.11.4 Attribute Binding

Attribute bindings are simple, powerful bindings. In an attribute binding, the directive represents a DOM attribute, while the source references an `Observable` View Model property. When the `Observable` View Model property updates, the View is automatically updated with the value of the property.

Example:

```
<div data-binding="{className: activeClass}"></div>
```

In this example, the DOM attribute "className" is bound to a property called `activeClass` in the View Model attached to the View.

Another example:

```
<img data-binding="{src: imageUrl}"></div>
```

If the source of an attribute binding is an `Observable`, any changes to the `Observable` will update the bound attribute.



## 22.11.5 Text Binding

Binding textual content to user interface elements is a very common scenario. To do so, the `@text` binding directive is provided. The `@text` directive binds the inner textual content of an element to a View Model property. Example:

```
<span data-binding="{@text: username}"></span>
```

Whenever the observable “username” property is set, the span element will be updated to reflect the content assigned to username.



The `@text` directive escapes HTML, including HTML entity codes. This helps prevent a class of attacks called Cross-Site Scripting (XSS for short) where a malicious user attempts to inject JavaScript into data fields that are then displayed verbatim (thus running code) on someone else’s machine.

## 22.11.6 Visibility Bindings

Visibility bindings are useful for controlling element visibilities. Here’s an example of a visibility binding, extending the previous sample:

```
<div data-binding="{@visible: showImage}">
  <img data-binding="{src: imageUrl}"></div>
</div>
```

Visibility bindings typically bind to Boolean values, but can also be bound to strings and collections.

If the source of a visibility binding is a string, it will resolve to the visible state if the string has more than 0 characters.

If the source of a visibility binding is a collection, it will resolve to the visible state if the collection has more than 0 items.

The visibility binding type has an inverse, called “`@hidden`”. Hidden uses the exact same logic as `@visible`, but inverted.

Example, using an ObservableCollection called “items”:

```
<div data-binding="{@hidden: items}">Sorry, no items exist.</div>
<div data-binding="{@visible: items}">
  <ul data-binding="{@source: items}">
    <!-- (items) -->
  </ul>
</div>
```

## 22.11.7 Events

Event bindings provide a way to declaratively wire up events. Event bindings take this form:

```
{@event - EVENT_NAME : HANDLER_NAME}
```

where `EVENT_NAME` is the name of a DOM event and `HANDLER_NAME` is a method in the View class.

## Example

```
<a href="javascript:void(0)" data-binding="{@event-onclick: handleClickLink}"></a>
```

In this figure, `onclick` follows the `@event-` portion of the directive. When the `onclick` event of the anchor element is raised, the method `handleClickLink` in the View class will be executed.

View event handlers take this form:

```
handleClickLink: function (event, element, context) {  
    alert("The link was clicked.");  
    return false;  
}
```

The parameters are as follows:

- `event`: The originating DOM event.
- `element`: The element in the binding.
- `context`: The data context (View Model) that the View is attached to.

You may wonder why the View Model ("context") of a data-bound event handler is passed into the method, when it will already be available in the View anyways through `this.ViewModel`. The reason for this will become apparent when dealing with collection bindings.

### 22.11.8 ObservableCollection

`ObservableCollection` extends the `Observable` concept to JavaScript Array collections.

An `ObservableCollection` wraps an internal array value and publishes events that notify consumers of changes to the collection.

When an `ObservableCollection` is updated, it will raise its binding event and any handlers attached will be executed and passed an instance of a `CollectionChangedEventArgs` object. This object represents a change that is about to be made to the actual collection. This object carries an operation type, such as `append`, `remove`, `clear`, etc., as well as a range representing which items in the array are to be modified.

`ObservableCollection` contains a number of methods for interacting with the underlying array, all of which publish the appropriate `CollectionChangedEventArgs` items.

The use of `ObservableCollection` facilitates one of the most powerful types of data binding: collection binding.

### 22.11.9 Collections

Collection binding is a powerful and versatile way to represent collections of items in a user interface.

Collection binding, also called "source binding", allows a View to bind to an `ObservableCollection` belonging to a View Model. Source-bound elements automatically update when the bound collection is modified.

To create a collection binding, the `@source` directive is used, with the source pointing to an `ObservableCollection` in the View Model.

Collection binding uses a template-based approach to generate the structure of the View. The source-bound element should contain a single child element that defines the template for each item in the collection. The child element does so by using the `@template-for` directive, with a source pointing to the same View Model member as the parent source-bound element.

## Example

```
<ul data-binding="{@source: customers}">
  <!-- This is the template item. -->
  <li data-binding="{@template-for: customers}">
    <div class="customer-listing">
      <a href="javascript:void(0)" data-binding="{innerHTML: displayName}"></a>
    </div>
  </li>
</ul>
```

When this View is attached to its View Model and added to the user interface, it will bind to the `customers` collection and populate the `<ul>` element with elements from the `<li>` template.

When a source-bound element is populated based on a collection, each item of that collection will serve as the View Model for a View instance whose markup is that of the template.

Event handlers specified in a template binding will still point to the parent View, but when the handler is executed, the `context` parameter will represent the View Model for that collection item.

Consider the following:

```
<div data-binding="{@source: items}">
  <div data-binding="{@template-for: items}">
    <a data-binding="{@event-onclick: handleClickItem}{innerHTML:
description}"></a>
  </div>
</div>
```

This piece of View markup is bound to a list of items in the View Model, and each item is displayed as an anchor element inside a `div`. Each item is associated with an `onclick` event handler called `handleClickItem`.

While this item will be bound to its own View Model (a member of `items`), its associated event handler `handleClickItem` will live in the original View, its parent View, and look like this:

```
handleClickItem: function (event, element, context) {
  // ... "context" will be the View Model whose bound View was clicked.
}
```



Template items in a source-bound element are created as Views themselves; therefore any valid binding expression can be used within them. Multiple levels of collection binding are possible. This is particularly useful when displaying hierarchical data.



If you wish to use source binding with TABLE elements to display rows or columns based on a collection, the binding expression must be attached to the `tbody` element, and not the `table` element.

```
<table>
  <tbody data-binding="{@source: features}">
    <tr data-binding="{@source: attributes}{@template-for: features}">
      <!-- etc. -->
    </tr>
  </tbody>
</table>
```



Source binding works on single `Observable` objects as well as `ObservableCollection` objects. Binding expressions in the template of an `Observable` source binding are resolved against the fields of the bound `Observable`.

## 22.12 HTML5 Viewer Embedding

### 22.12.1 About HTML5 Viewer Embedding

There are two ways to embed the Geocortex Viewer for HTML5:

- [Embed an HTML5 Viewer within an iframe.](#)  
This method is the simplest way to place a viewer on an existing web page. You can only customize the viewer as far as configuring the viewer configuration files.
- [Embed an HTML5 Viewer within the entire page.](#)  
This advanced method creates a custom HTML5 Viewer web application that makes up the entire web page. This method offers superior programmatic customization of the viewer, of both functionality and appearance.

## 22.12.1.1 Architecture

Before embedding the Geocortex Viewer for HTML5 (GVH) within the entire web page, you should be familiar with the following GVH loading components:

- `module geocortex`
  - `module framework`
    - `module application`
      - `class ApplicationLoader`: A base class that uses a `ResourceSet` to efficiently load scripts and style sheets before creating and initializing a framework `Application`.
      - `interface ApplicationInitializationOptions`: Options used to by the `ApplicationLoader` class to initialize a framework `Application`.
      - `class CaselessMap`: An object that has methods for accessing its properties in a case-insensitive way.
      - `module resources`: This module defines the notion of loadable resource objects and encapsulates everything directly related to loading dependencies on the fly.
        - `class Resource`: A base class that represents anything that is loadable.
        - `class ScriptResource extends Resource`: A script resource to be loaded once.
        - `class StyleResource extends Resource`: A style sheet resource to be loaded once.
        - `class ResourceSet extends Resource`: A named collection of resource objects that is nestable, since a `ResourceSet` itself is a `Resource`. All content nested within a `ResourceSet` can be loaded recursively.
        - `enum ResourceStatus`: An enumeration that includes the valid loading statuses of a resource.
  - `module essentialsHtmlViewer`
    - `class Splash`: Controls the behavior of the splash screen.
    - `class ViewerLoader extends ApplicationLoader`: This class provides the means to create and load `ResourceSet` and `ViewerApplication` objects.
    - `interface ViewerInitializationOptions`: Options used by the `ViewerLoader` class to initialize a `ViewerApplication`.

## 22.12.1.2 ViewerLoader

The `ViewerLoader` class is responsible for defining and loading the viewer's `ResourceSet`, and then initializing the `ViewerApplication`.

The `ViewerLoader` constructor accepts a `ViewerLoaderOptions` object with the following property:

- `loaderResourcePrefix?`: An optional string to prefix to the URL of each local resource. This allows you to rebase the viewer's resources relative to the viewer. Defaults to `"/"` if not specified.



As of HTML5 Viewer 2.5.1, the `baseUr1` property has been deprecated; use `loaderResourcePrefix` instead.

The `ViewerLoader` constructor then generates the viewer's default `ResourceSet`, exposing it on the new `ViewerLoader` instance as a property named `resourceSet` so that it may be customized prior to loading if necessary.

### 22.12.1.3 Resource

A resource represents anything that is loadable, including `ScriptResource`, `StyleResource` and `ResourceSet` objects. `ScriptResource` and `StyleResource` objects are named by the URLs provided to define their resources, while `ResourceSet` objects are explicitly named.

All resource objects have a `load()` method, which accepts a `ResourceLoadOptions` object with the following properties:

- **onLoaded?** (Function): An optional callback function that runs when the resource has successfully finished loading. If the resource is a script, the callback function runs when the script has finished executing. This resource is passed as an argument to the callback function.



`StyleResource` objects call their `onLoaded` callback function prematurely due to cross-browser limitations of detecting when a style sheet has finished loading, however, CSS priority will be maintained.

- **onError?** (Function): An optional callback function that runs when the resource has failed to load. This resource is passed as the first argument, and an `Error` instance as the second argument to the callback function.

### 22.12.1.4 ResourceSet

A `ResourceSet` object is a named collection of loadable resource objects. A `ResourceSet` is infinitely nestable; in other words, `ResourceSet` objects may contain other `ResourceSet` objects. A `ViewerLoader` instance generates a default `ResourceSet` tree, which contains all the GVH dependencies as resource objects, exposed as a property named `resourceSet`.

As a `ResourceSet` is a type of `Resource`, its entire contents can be loaded recursively via the `load()` method. Other `ResourceSet` methods include:

- **find(string)**: Loops through the `ResourceSet` and returns a resource whose name matches the specified string. It will return the current `ResourceSet` if it matches.
- **append(ResourceItem[])**: Interprets the items specified as `Resource` objects, and adds them to the end of the `ResourceSet`'s collection of `Resource` objects.
- **prepend(ResourceItem[])**: Interprets the items specified as `Resource` objects, and adds them to the beginning of the `ResourceSet`'s collection of `Resource` objects.



A `ResourceItem` object can be any `Resource` object or anything that can be automatically interpreted as a `Resource` object. An object literal is interpreted as a `ResourceSet` object. A string ending in `.js` is interpreted as a `ScriptResource` object. A string ending in `.css` is interpreted as a `StyleResource`. If your script or CSS file name has an unconventional ending, you must explicitly create the desired type of resource to supply it as a `ResourceItem`. For example, `new ScriptResource("my-script.txt")`.

The default GVH `ResourceSet` tree includes the following mutable `ResourceSet` objects:

- `everything`
  - `dependencies`
    - `css`
    - `tools`
    - `esri`
- `viewer`

## 22.12.2 Embed an HTML5 Viewer within an Iframe

An `iframe` (inline frame) element allows you to place an HTML document within a frame; for example, the `Index.html` file that hosts the HTML5 Viewer.

### 22.12.2.1 Basic Example

To embed the HTML5 Viewer within an `<iframe>` element in your HTML, add the following within your `<body>` tag:

```
<iframe style="width:800px; height:600px;"
src="http://MyServer.com/Html5Viewer/Index.html">
  <p>Your browser does not support iframes.</p>
</iframe>
```

Replace the value of the `src` attribute with the URL to your HTML5 Viewer. To modify the size of the `<iframe>`, change the values of the `width` and `height` attributes.



You can include an optional message within the `<iframe>` tag that only displays if the user's browser does not support iframes.

### 22.12.3 Embed an HTML5 Viewer within the Entire Page

This section assumes your familiarity with the structure of an HTML file.

For more information about HTML, see [1.9 A quick introduction to HTML](#) from the W3C's HTML5 specification.

### 22.12.3.1 Basic Example

The most basic way to embed a Geocortex Viewer for HTML5 within the entire page is to edit your HTML and add the following `<script>` tags just before the end of your `</body>` tag:

```
<script src="Resources/Compiled/loader.js"></script>
<script>
  var viewerConfig = {
    "configurations": {
      "default": "Resources/Config/Default/" + shellName + ".json.js"
    }
  };
  new geocortex.essentialsHtmlViewer.ViewerLoader().loadAndInitialize();
</script>
```

The first `<script>` tag loads the necessary `loader.js` script.

The second `<script>` tag defines the `viewerConfig` and creates a new default instance of a `ViewerApplication` object. The viewer dependencies and resources are loaded.

### 22.12.3.2 Customized Example

You can customize the `ViewerApplication` object created, as the `loadAndInitialize()` method accepts a `ViewerInitializationOptions` parameter:

```
<script src="Resources/Compiled/loader.js"></script>
<script>
  var viewerConfig = {
    "configurations": {
      "default": "Resources/Config/Default/" + shellName + ".json.js"
    }
  };
  new geocortex.essentialsHtmlViewer.ViewerLoader().loadAndInitialize({
    hostElement: document.body,
    configBase: "Resources/Config/Default/",
    onInitialized: function(viewer, viewerLoader) {
      console.debug("Viewer Initialized:", viewer);
    }
  });
</script>
```

The `hostElement` is the HTML element that hosts the viewer. The `configBase` is the location of the viewer configuration files.

The newly-created `ViewerApplication` object and the `ViewerLoader` object are passed as parameters to the `onInitialized` callback function when the `ViewerLoader`'s `ResourceSet` finishes loading and the `ViewerApplication` is initialized. In the above example, the callback function outputs the new viewer to the console.



## 22.12.3.3 Viewer Initialization Options


To customize the viewer created, the `loadAndInitialize()` method accepts a `ViewerInitializationOptions` parameter with the following optional properties, the first three of which are inherited from the `ApplicationInitializationOptions` base class:


- **hostElement?** (`HTMLElement`): The DOM element to host the application. The default is `document.body`.
- **onInitialized?** (`((application: Application, loader: ApplicationLoader) => void)`): A callback function that is called when the `ApplicationLoader`'s `ResourceSet` has loaded and the `Application` has been initialized. The `Application` and `ApplicationLoader` objects are passed as parameters to the callback function.



`Application` and `ApplicationLoader` are the base classes of `ViewerApplication` and `ViewerLoader`, respectively.

- **onError?** (`(error: Error, loader: ApplicationLoader, application?: Application) => void`): A callback function that is called when the `ApplicationLoader`'s `ResourceSet` fails to load, or the `Application` fails to initialize. The `Error`, `ApplicationLoader` and `Application` objects are passed as parameters to the callback function.
- **debug?** (`boolean`): Whether or not to output debug information to the console (whether `debugMode` should be set on the `ViewerApplication` instance). The default is `false`.
- **aliases?** (`{ [viewerName: string]: string }`): An object whose property names represent aliases, and whose property values represent corresponding viewer URLs. To specify an alias, use the [viewer URL parameter](#). The default is an empty object.
- **offline?** (`boolean`): Whether or not the viewer should start in offline mode. The default is `false`.
- **configBase?** (`string`): Specifies the folder which contains the viewer configuration files. The trailing slash will be added if missing. The default is `"Resources/Config/Default/"`.  
If you specify a `configBase` URI, ensure a `ViewerSettings.json.js` exists in the same directory where the HTML with an embedded viewer lives.
- **shell?** (`string`): Specifies which shell to use. Possible values include: `Desktop`, `Handheld` and `Tablet`. If omitted, the default shell is automatically determined by detecting the user agent.
- **viewerConfigUri?** (`string`): Specifies the location of a particular viewer configuration file to load. This parameter overrides the `configBase` and `shell` parameters. The default is `configBase + shell + ".json.js"`.
- **query?** (`geocortex.framework.application.loading.CaselessMap`): A `CaselessMap` of query parameters. The default is the query string.
- **onSiteInitialized?** (`((viewer: ViewerApplication, loader: ViewerLoader) => void)`): A callback function that is called when the newly-created `ViewerApplication` object fires its `SiteInitializedEvent` framework event. The `ViewerApplication` object and the `ViewerLoader` object are passed as parameters to the callback function.

 The following options can be overridden by adding query string parameters of the same name to the viewer URL: `debug`, `configBase`, `viewerConfigUri`.

 When adding query string parameters to the viewer URL, a question mark (?) should precede the first query parameter, and an ampersand (&) should precede subsequent query parameters. For example, `http://MyUrl.com/index.html?configBase=MyConfigFolder&debug=true` would set the folder containing viewer configuration files to `MyConfigFolder` and start the viewer in debug mode.

 All option names are case-insensitive.

#### 22.12.3.4 Splash Screen

We recommend you include the optional splash screen. The `ViewerLoader` will automatically close the splash screen when a viewer is initialized.

To add a splash screen:


1. Use either the **Simple Method** or **Recommended Method** to add the splash screen styles:

- **Simple Method:** In your HTML, add the following within your `<head>` tag:

```
<link rel="stylesheet" href="Resources/Styles/splash.css"/>
```

- **Recommended Method:** In your HTML, within your `<head>` tag, create a `<style>` element containing the entire contents of `Resources/Styles/splash.css`:

```
<style>
  .splash-overlay, .splash-overlay * { margin: 0; padding: 0; }
  ... (Truncated for the sake of brevity, see splash.css for the rest)...
</style>
```

 This method of including the splash screen will reduce the number of web requests so your website will load slightly faster.

2. In your HTML, just after your `<body>` tag, add the following:

```
<div class="splash-overlay">
  <div class="splash-plate">
    
    <p class="splash-paragraph">This application uses licensed Geocortex
    Essentials technology for the Esri<sup>&reg;</sup> ArcGIS platform. All rights
    reserved.</p>
  </div>
</div>
```

#### 22.12.3.5 Customize the ResourceSet

You can modify the default [ResourceSet](#) of a `ViewerLoader` object via its `resourceSet` property before initializing a viewer.

For example, to add a style sheet named `my-styles.css` to the beginning of the `css` resource set, add the following within a `<script>` tag:

```
var loader = new geocortex.essentialsHtmlViewer.ViewerLoader();
loader.resourceSet.find("css").prepend(["my-styles.css"]);
```

Similarly, to add two scripts to the beginning and end of the `jquery` resource set:

```
loader.resourceSet.find("jquery")
  .prepend(["my-first-plugin.jquery.min.js"])
  .append(["my-last-plugin.jquery.min.js"]);
```

To create and add a new `ResourceSet` named `my-library` to the end of the `dependencies` resource set:

```
var ResourceSet = geocortex.framework.application.resources.ResourceSet;

var myLibraryResourceSet = new ResourceSet("my-library", [
  "my-library-styles.css",
  "my-library-basics.js",
  "my-library-plugin.js"
]);

loader.resourceSet.find("dependencies").append([myLibraryResourceSet]);
```

The `ResourceSet` will automatically assume the `Resource` type based on the file extension. If the file extension is different or missing, you can explicitly create the `Resource` instance instead:

```
var StyleResource = geocortex.framework.application.resources.StyleResource;
var ScriptResource = geocortex.framework.application.resources.ScriptResource;

var myLibraryResourceSet = new ResourceSet("my-library", [
    new StyleResource("my-library-styles"),
    new ScriptResource("my-library-basics"),
    new ScriptResource("my-library-plugin")
]);

loader.resourceSet.find("dependencies").append([myLibraryResourceSet]);
```

Finally, to create the viewer:

```
loader.loadAndInitialize();
```

---

## Appendix A: Glossary

**ArcGIS Server.** Esri's software that creates GIS services over the web for web-mapping applications.

**Domain Name.** The part of a URI or URL that specifies the Internet address of the web server. For example, in `http://services.arcgisonline.com/arcgis/rest/services/`, the domain name is `services.arcgisonline.com`.

**Epoch Time.** See [Unix Time](#).

**Fully Qualified URI/URL.** A URI or URL that specifies all the parts of the domain name and is unambiguous in any context. For example, `http://myserver.mycompany.com/myfolder/myfile.html`. The following URL is not fully qualified: `http://myserver/myfolder/myfile.html`, and neither is this: `http://localhost/myfolder/myfile.html`. Relative URLs are also not fully qualified: `myfolder/myfile.html`.

**GIS.** Geographic Information System, a system that captures, analyzes or manages data that is linked to a location on a map.

**IIS.** Internet Information Services is Microsoft's web server software.

**KML.** Keyhole Markup Language - a schema for rendering geographic markup on web-based maps. Based on XML. See also: [OGC](#).

**Map.** The maps referred to in this application are web-based maps, which are fundamentally different from paper-based maps in that they are both interactive and searchable. Web maps contain data in many forms, which can be searched, annotated, and used for analysis and decision-making.

Web maps are also referred to as **basemaps** and **operational maps**. Basemaps supply background and contextual information in layers. Basemaps usually contain information about features or structures that do not change often like highways, rivers, mountains, and borders. Operational maps often contain layers with additional data used for specific tasks like tracking, research, or analysis. The data in operational maps is usually within a specific area of interest, for example forestry, population, or earthquake incidents, and are used to highlight quantities, densities, or distribution. Operational layers are usually added to basemaps, which provide a contextual background for the operational information.

**Map Service.** A map service is how most maps are published over the Internet. Maps are generated by a map server using data from a GIS database. Well-known map services are provided by ArcGIS, Google Maps, Bing, and MapQuest but there are many others. Map services use Global Positioning System (GPS) data to describe the physical locations of features. Map services can be used by many different client applications. Tools and specifications such as Keyhole Markup Language (KML) and Open Spatial Consortium (OGC) have made it easier for applications to use map services by providing common languages and standards for rendering maps.

A **dynamic** map service is where the server draws maps on demand, which means that the map is re-drawn each time the user zooms or pans. A **cached** map service, is a set of tiled map images that are pre-rendered so that they display rapidly. Cached maps are created at specified scale levels and stored on a server (server-side cache) or locally. Cached maps can be a higher quality and use features like 3D and transparency.

**Modules.** Are packages of functionality that can be independently developed, tested, and deployed. In many situations, modules are developed and maintained by separate teams. A typical application is built from multiple modules. Modules can be used to represent specific business functions. Modules also encapsulate common application infrastructure or services (for example, logging and exception management services) that can be reused across multiple applications.

**OGC.** The Open Geospatial Consortium is an international organization that sets standards for geospatial services and content, data sharing and GIS data processing. They have issued over 30 standards including:

- **GML** - Geography Markup Language: XML-format for geographical information.
- **KML** - Keyhole Markup Language: XML-based language schema for expressing geographic annotation and visualization.
- **WFS** - Web Feature Service: describes a service for the discovery, querying of, and operations for the transformation of data.
- **WMS** - Web Map Service: a standard protocol for serving dynamic geographical map images over the Internet that are generated by a map server.
- **WMTS** - Web Map Tile Service: a standard for implementing servers to deliver tiled maps.

**QR Code.** A Quick Response Code is a matrix barcode that can be rapidly decoded. It is often used to encode URLs to a website. A mobile device with an integrated camera and QR Code software can scan the encoded URL and open the web page immediately.

**Regions.** Regions are logical placeholders defined within the application's UI (in the shell or within views) in which Views are displayed. Regions allow the layout of the application's UI to be updated without requiring changes to the application logic. Typically, `<div>` elements are used as regions to automatically display Views. However, any HTML element that can host content can be used as a region. Views can be displayed within a Region programmatically or automatically. Regions can be located by other components through the `ViewManager`.

**REST.** Representational State Transfer provides a simple, open web interface to services hosted by ArcGIS Server. All the resources and operations exposed by the REST API are accessible through a hierarchy of endpoints or Uniform Resource Locators (URLs) for each GIS service published with ArcGIS Server.

**Shell.** The Shell defines the overall layout and structure of the application. It is usually not aware of which modules it hosts. It usually implements common application services and infrastructure, but most of the application's functionality and content is implemented within the modules. The Shell also provides the top-level window or visual element that hosts the different UI components provided by the loaded modules.

**String.** A series of characters, consisting of letters, numbers or symbols. A string can also be empty or null (not assigned). A string is a very common data type.

**Unix Time.** The number of milliseconds since the Unix epoch: January 1, 1970 at midnight UTC. Unix time is a common way for software to store times internally. To convert between human-readable times and Unix time, use a converter like <http://www.epochconverter.com/>.

**URI.** URIs (Uniform Resource Identifiers) identify and locate resources on a network. The network can be an intranet or the Internet. A URL such as `http://sampleserver1.geocortex.com/arcgis/rest/services/` is one type of URI. A file path like `..\..\Resources\Styles\Common.css` is another type of URI.

**URL.** URLs (Uniform Resource Locators) are also known as web addresses. They define the location of websites, web pages, and other resources on a network. For example, `http://www.esri.com/` is a URL. The network can be an intranet (a local network) or the Internet.

**URL Parameter.** A string that you attach to the end of a URL to specify initial values and actions. For example, the HTML5 viewer's `viewerConfigUri` parameter is used to specify the location of the configuration file to load. The `runWorkflow` URL parameter runs the specified workflows when the viewer is launched. URL parameters are also known as HTTP query strings.

**UTC.** Closely related to Greenwich Mean Time, Coordinated Universal Time (UTC) is the primary time standard by which clocks and time are regulated. Viewed as a time zone, the United Kingdom is UTC+0, New York is UTC-05:00 (subtract 5 hours from UTC time), and the Netherlands is UTC+01:00 (add one hour to UTC time).

**UTM.** Universal Transverse Mercator geographic coordinate system. A transverse Mercator projection orients the equator north-south through the poles, providing a north-south swath with little distortion. The orientation of the cylinder onto which the map is projected, is changed slightly for each swath. This creates relatively undistorted regions. Each swath is called a UTM zone and is 6 degrees of longitude wide.

**Views.** Views are UI controls that encapsulate the UI for a particular feature or functional area of the application. Views are used in conjunction with the Model-View-ViewModel (MVVM) or Model-View-Presenter (MVP) patterns, which are used to provide a clean separation of concerns between the application's presentation logic (UI) and business logic. Views encapsulate the UI and define user interaction behavior, allowing the View to be updated or replaced independently of the underlying application functionality. Views use data binding to interact with the View Model and presenter classes.

**WFS.** Web Feature Service: describes a service for the discovery, querying of, and operations for the transformation of data.

**Widget:** An interactive element that is created and controlled through scripting. Any controls that is not native to HTML, or that is greatly enhanced by scripting, is a widget. For example, sliders, fly-out menus, tree systems, and drag-and-drop controls are widgets.

**WMS.** Web Map Service: a widely supported format for web-based maps and a standard issued by the OGC on the implementation of dynamic map services. See also: [OGC](#).

**WMTS.** Web Map Tile Service: a standard for implementing servers to deliver tiled maps.

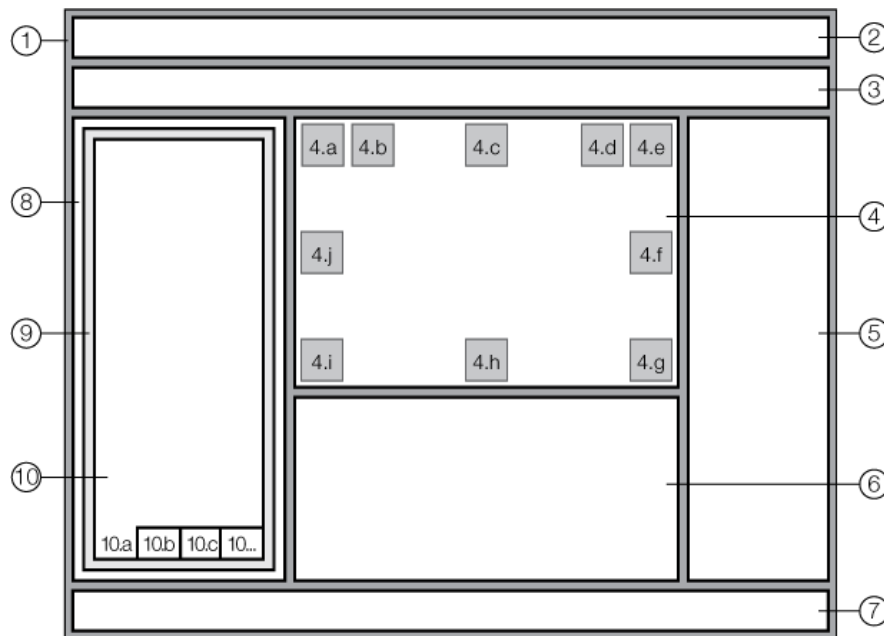
## Appendix B: Regions

Regions are areas in an interface where views can be activated. Each view has a `region` property that controls which region the view is activated in. This is where the view is displayed.

For example, in the Desktop and Tablet interfaces, the Banner Module has a view called `BannerView` that displays the banner. `BannerView`'s `region` property is set to `HeaderRegion` by default. This makes the banner display in the `HeaderRegion`.

### B.1 Regions in the Desktop and Tablet Interfaces

The diagram below shows the main regions used in the Desktop and Tablet interfaces.



**Main regions in the Desktop and Tablet interfaces**

1. **ApplicationRegion**: Contains all the other regions. The `ApplicationRegion` is used by the [Log Module](#) and [Shells Module](#).
2. **HeaderRegion**: Holds the [banner](#).
3. **ToolBarRegion**: Holds the [tabbed toolbar](#).
4. **MapRegion**: Holds the [map](#). The following regions lie on top of the `MapRegion`. These regions are used by map widgets, like the `Basemap Switcher` and `Offline indicator`. You can also use these regions to display messages to the user, or anything else that you want to appear on top of the map.
  - a. **TopLeftMapRegion**
  - b. **NavigationMapRegion**
  - c. **TopCenterMapRegion**
  - d. **BasemapMapRegion**



- e. `TopRightMapRegion`
  - f. `MiddleRightMapRegion`
  - g. `BottomRightMapRegion`
  - h. `BottomCenterMapRegion`
  - i. `BottomLeftMapRegion`
  - j. `MiddleLeftMapRegion`
5. `RightPanelRegion`
  6. `ResultsRegion`: Holds the [Results Table](#) and other results-related views, like [feature details](#). `ResultsRegion` is in the `BottomPanelRegion` (not shown in the diagram).
  7. `FooterRegion`: Holds the application's [footer](#). The `FooterRegion` has two regions on top of it—`LeftFooterRegion` and `RightFooterRegion`. The `LeftFooterRegion`'s content is left justified. The `RightFooterRegion`'s content is right justified.
  8. `LeftPanelRegion`: Has a view-container view inside it that holds the `DataRegion` (9).
  9. `DataRegion`: Hosts a number of container regions, listed below (10). The `DataRegion` is used by the [Workflow Module](#).



When a view is placed directly into the `DataRegion`, there is a tab for the view. This is useful for running workflows—you can display workflow content in one or more tabs so the user can navigate easily between the workflow, map layers, and other `DataRegion` tabs.

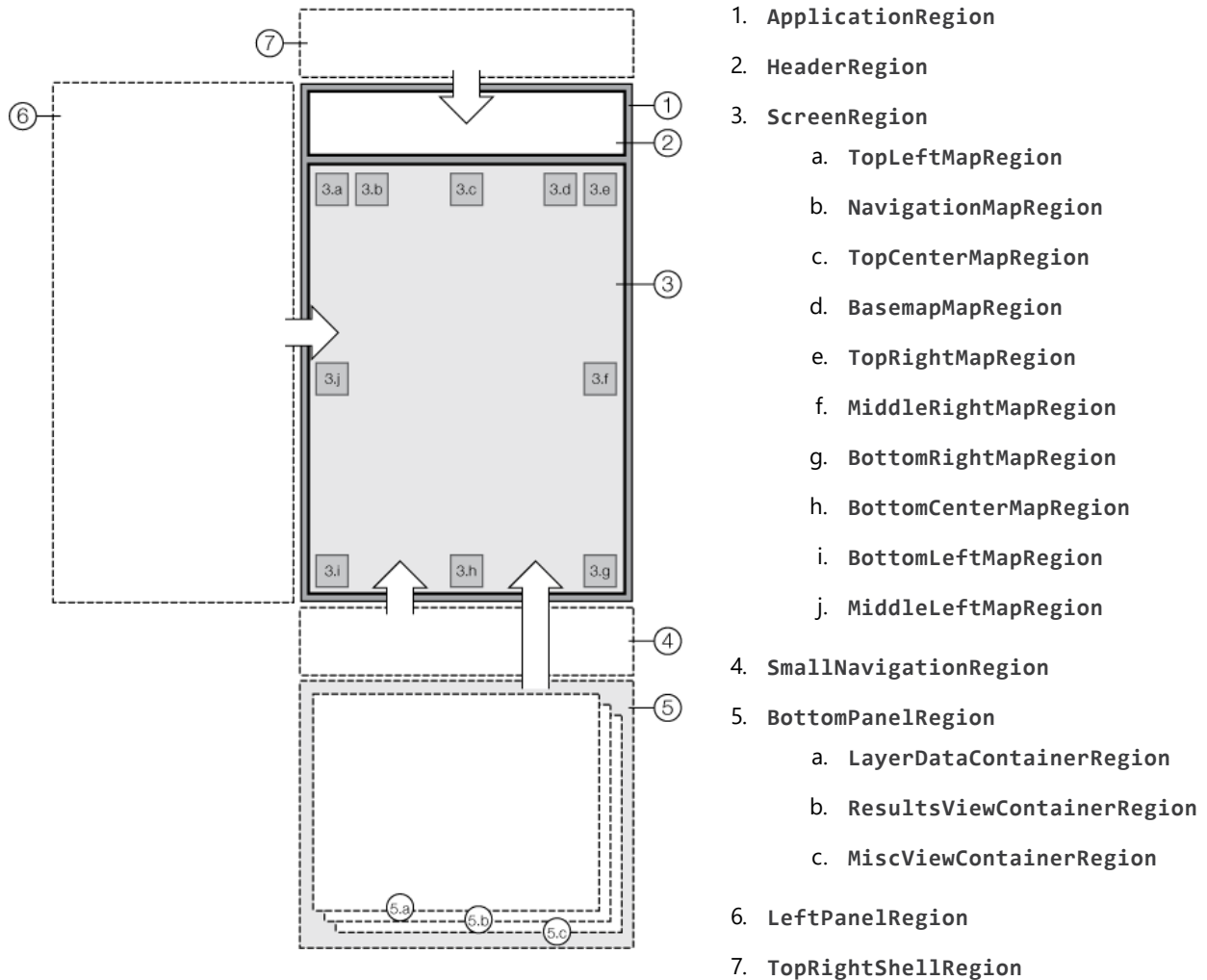
10. **Container Regions:** The following regions lie on top of the `DataRegion`. These container regions are where custom views are usually hosted.
  - a. `HomePanelContainerRegion`: The default region for the [Home Panel](#).
  - b. `LayerDataContainerRegion`: The default region for the [Layer List](#) and other layer-related views.
  - c. `DataFrameResultsContainerRegion`: The default region for the [Results List](#) and other results-related views, like [feature details](#).
  - d. `FeatureEditingContainerRegion`: Holds [editing](#)-related views.
  - e. `SimpleQueryBuilderContainerRegion`: Holds the Simple [Query Builder](#).
  - f. `SimpleFilterBuilderContainerRegion`: Holds the Simple [Filter Builder](#).

The `BottomPanelRegion` is not shown in the diagram. `BottomPanelRegion` hosts a container region that can display multiple side-by-side panes containing [charts](#) or [third-party applications](#). `BottomPanelRegion` also contains the `ResultsRegion`.

The `ModalWindowRegion` is also not shown in the diagram. `ModalWindowRegion` is used to display content that the user must acknowledge before proceeding with the map session. For example, alerts and confirmation messages are often displayed in the `ModalWindowRegion`.

## B.2 Regions in the Handheld Interface

The Handheld interface has the regions shown in the figure below, plus one more: `ModalWindowRegion`. The `ModalWindowRegion` is used to display content that the user must acknowledge before proceeding with the map session. For example, alerts and confirmation messages are often displayed in the `ModalWindowRegion`.



### Regions in the Handheld interface

The **ApplicationRegion** hosts the entire application.

The **HeaderRegion** hosts the I Want To menu button, Global Search box and Tools button.

The **ScreenRegion** hosts the map, and contains multiple subregions that correspond to different positions on the map.

The **SmallNavigationRegion** hosts the navigation bar that appears at the bottom of the screen when the user clicks



at the bottom right corner of the screen.

The **BottomPanelRegion** hosts several view containers, each of which holds a region. These container regions are where custom views are most likely to be added, particularly the **MiscViewContainerRegion**. The regions are:

- **LayerDataContainerRegion:** Holds the layer list and other layer-related views.
- **ResultsViewContainerRegion:** Holds the Results List, Feature Details, and other results-related views.
- **MiscViewContainerRegion:** Holds various other views, such as the Home Panel.

The `LeftPanelRegion` hosts the I Want To menu.

The `TopRightShellRegion` hosts the Compact Toolbar.

The Handheld, Tablet, and Desktop interfaces all display the map in the `MapView`. In the Handheld interface, the `MapView` uses the `ScreenRegion`—there is no dedicated `MapRegion` like there is in the Desktop and Tablet interfaces. When the `MapView` is active in the `ScreenRegion`, you can use all the same regions that lie on top of the `MapRegion` in the Desktop and Tablet interfaces. You can use these regions to display map widgets, messages to the user, or anything else that you want to appear on top of the map.

## Appendix C: File Locations

The table below lists the locations of the files and folders for an HTML5 viewer that was installed using Essentials.

Curly brackets represent a file path. For example, {GE\_INSTALL} is the path of the Essentials installation folder.

Square brackets represent the name of the particular item. For example, in {GE\_HOME}\Sites\[site], [site] stands for the name of a particular site.

### Default File and Folder Locations

File or Folder	Shorthand	Default Location
Geocortex Essentials installation folder	{GE_INSTALL}	C:\Program Files (x86)\Latitude Geographics\Geocortex Essentials
Geocortex Essentials REST	{GE_HOME}	Unnamed instance of Essentials: {GE_INSTALL}\Default\REST Elements Named instances of Essentials: {GE_INSTALL}\[instance]\REST Elements
Sites folder	{GE_SITES}	{GE_HOME}\Sites
Tile package (TPK) files folder		{GE_SITES}\TPKs
Folder for a particular site		{GE_SITES}\[site]
Site configuration file		{GE_SITES}\[site]\Site.xml
Geocortex viewers	{G_VIEWERS}	{GE_SITES}\[site]\Viewers
Viewers		{G_VIEWERS}\[viewer]
Viewer virtual directory		{G_VIEWERS}\[viewer]\VirtualDirectory
HTML5 viewer configuration files		{G_VIEWERS}\[viewer]\VirtualDirectory\Resources\Config\Default
HTML5 viewer custom images		{G_VIEWERS}\[viewer]\VirtualDirectory\Resources\Images\Custom
HTML5 viewer styles		{G_VIEWERS}\[viewer]\VirtualDirectory\Resources\Styles\Custom

File or Folder	Shorthand	Default Location
HTML5 viewer language files		C:\inetpub\wwwroot\[viewer]\Resources\Locales

## Appendix D: Command Reference



As of HTML5 Viewer 2.5, the Command Reference has been replaced by the Geocortex SDK for HTML5 API Reference. For more information, see [Geocortex SDK for HTML5 API Reference on page 422](#).

## Appendix E: Event Reference



As of HTML5 Viewer 2.5, the Event Reference has been replaced by the Geocortex SDK for HTML5 API Reference. For more information, see [Geocortex SDK for HTML5 API Reference on page 422](#).

## Appendix F: State Reference

The following tables list HTML5 Viewer application states. Only a single global state may be active at any time, whereas any number of non-global states may be active simultaneously.

<b>F.1 Global States</b> .....	<b>452</b>
<b>F.2 Non-global States</b> .....	<b>452</b>

### F.1 Global States

State Name	Description
DrawMarkupState	Triggered when any tools that involve drawing markup on the map are active.
FeaturePlacementState	Triggered when the feature editor is started by choosing a template.
FindDataState	Triggered when identification-based tools are active.
IdentifyState	Triggered when the default standalone identification tool is active. This state overrides FindDataState.
MeasureState	Triggered when any tools that involve performing measurements on the map are active.
SelectMarkupForEditingState	Triggered when the tool to select markup on the map for editing is active.

### F.2 Non-global States

State Name	Description
DefaultState	The default state of the application.
EditorActiveState	Active when the View/Edit Attributes view is active.
EditingMarkupState	Triggered when a piece of markup is being edited.
EditingMeasurementMarkupState	Triggered when a piece of measurement markup is being edited.
FeaturePlacementGraphicState	Triggered when the feature editor's graphic editing state is started.
FeaturePlacementPointGraphicState	Triggered when the feature editor's graphic editing state is started for a point-based graphic.
FindDataBufferingState	Triggered when buffering is enabled for the markup drawing tool.



State Name	Description
IdentifyBufferingState	Triggered when buffering is enabled for the standalone identification tool.
SnappingState	Triggered when snapping is enabled.
TransientActiveState	Is active whenever any context-sensitive toolbar (also known as a transient element) is active.